Algoritmeja autonomiseen avaruustutkimukseen

Jarkko Huovila

Tampereen yliopisto
Informaatiotieteiden yksikkö
Tietojenkäsittelyoppi
Pro gradu -tutkielma
Ohjaaja: Erkki Mäkinen
Kesäkuu 2015

Tampereen yliopisto
Informaatiotieteiden yksikkö
Tietojenkäsittelyoppi
Jarkko Huovila: Algoritmeja autonomiseen avaruustutkimukseen
Pro gradu -tutkielma, 66 sivua
Kesäkuu 2015

Avainsanat: Avaruustutkimus, ANTS, autonomia, optimointi, parviäly, tekoäly

Tiivistelmä

Avaruus on monin tavoin mielenkiintoinen tutkimuskohde, jonka tutkimiseen panostetaan koko ajan enemmän resursseja. Tässä tutkielmassa perehdytään tekoälyn tuomiin mahdollisuuksiin. Autonomisen tutkimuksen suorittaminen avaruusaluksin ja satelliitein vaatii paljon suunnittelua ja ohjelmointia. Uusilla tekoälyn ohjelmointimenetelmillä, kuten geneettisillä algoritmeilla, voidaan päästä lähemmäs tavoiteltua autonomiaa avaruuteen suuntautuvissa tutkimusprojekteissa.

Keskityn tutkielmassani käsittelemään avaruusalusten tai satelliittien parvia, jotka yhteistyöllä suorittavat vaativia tutkimuksia. Parven tutkimustyön koordinointi ja hallinta vaativat paljon suunnittelua ja optimointia, sillä parven jäsenillä on omat erikoistehtävänsä. Tästä johtuen joudutaan ratkomaan erilaisia ristiriitatilanteita mm. tutkimuksen suorittamisen ja parven turvallisuuden suhteen. Lisäksi optimoinnissa pitää huomioida käytössä olevat resurssit ajan, energian ja polttoaineen suhteen.

Yksi mielenkiintoisimmista projekteista on Autonomous Nano Technology Swarm (ANTS) –arkkitehtuuri, jonka avulla voidaan esimerkiksi kartoittaa asteroidivyöhykkeen asteroideja. Parvi voi sisältää satoja pieniä tutkimussatelliitteja, ja jotta kaikki sujuisi suunnitelmien mukaan, täytyy parven toimia mahdollisimman autonomisesti ja älykkäästi. Tämän toiminnan mahdollistamiseksi on tekoälymenetelmillä kehitetty uusia ja tehokkaita algoritmeja.

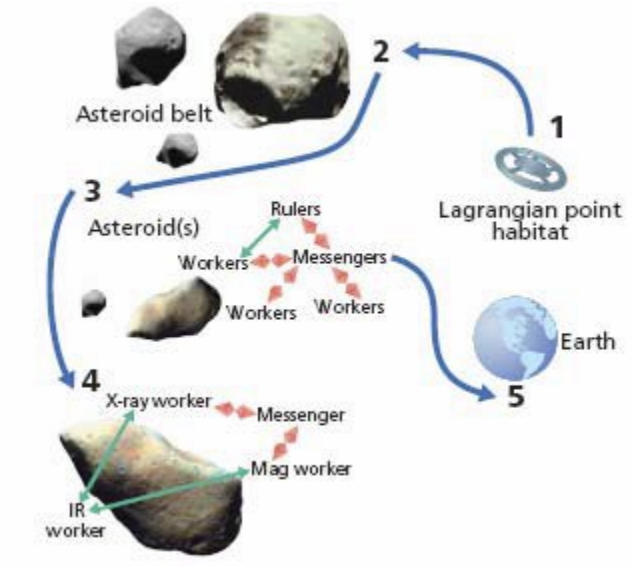
Sisällys

1. Johdanto

Tässä tutkielmassa perehdytään autonomisten joukkojen liikkumiseen avaruudessa erilaisissa tutkimus- ja pelastustehtävissä. Joukko voi muodostua kahdesta tai useammasta koneesta kuten satelliiteista tai avaruusaluksista. Suurimmissa tutkimustehtävissä voi olla käytettävissä satoja pieniä satelliitteja tai avaruusaluksia. Perusideana on joukon autonomisuus sekä liikkuminen ennalta määrätyssä muodostelmassa. Autonomiaan pyritään erilaisin tekoälymenetelmin, jotta ihmisten suorittamaa valvontaa ja ohjausta tarvittaisiin mahdollisimman vähän tai ei ollenkaan. Muodostelmassa liikkumisen suunnittelun tärkeimpiä tekijöitä on lentoradan optimointi matkan pituuden, polttoaineen kulutuksen ja turvallisuuden kannalta.

Avaruustutkimuksessa ollaan hiljalleen siirtymässä Maan kiertoradalta kauemmas avaruudessa tapahtuviin tutkimusprojekteihin, jolloin sekä turvallisuus että taloudelliset tekijät pakottavat etsimään uusia ratkaisuja miehitettyjen avaruuslentojen rinnalle. Miehitettyjen avaruuslentojen ongelmia ovat mm. suuret kulut, ihmisten mukanaolon välttämättömyys ja tietty joustamattomuus, joka ei salli suuria muutoksia tutkimussuunnitelmaan. Miehitetty avaruuslento vaatii paljon tutkimusta ja testausta, koska epäonnistuminen esimerkiksi laukaisussa tuhoaa koko projektin yhdellä kertaa. Kun käytetään pieniä ja halpoja yksiköitä, ei yhden tuhoutuminen tee vielä suurta vahinkoa koko projektille, ja joissain tapauksissa tuhoutuneen koneen tilalle voidaan jopa suhteellisen helposti lähettää korvaava yksikkö.

Autonomian saavuttamiseksi tehdään tutkimustyötä tekoälyn alueella. Tutkimuskohteina ovat autonominen laskenta, evolutionääriset ja geneettiset algoritmit sekä parviäly. Autonomisen järjestelmän tavoitteena on pystyä huolehtimaan itse omista toiminnoistaan, ylläpidosta ja korjauksista [Kephart and Chess, 2003]. Muodostelmassa liikkumista varten kehitellään erilaisia optimointialgoritmeja parhaan polun löytämiseksi sekä törmäysten välttämiseksi. Muodostelmassa liikkuminen synnyttää aina ristiriidassa olevia tavoitteita eri yksiköiden välille. Tällaisia tavoitteita ovat yksilön polttoaineen kulutuksen minimointi ja koko joukon polttoaineen kulutuksen tasapainottaminen, jotta joukko pysyisi mahdollisimman pitkään toimintakykyisenä. Ristiriitoja aiheuttavat myös lyhimmän ja nopeimman tai toisaalta turvallisen ja törmäyksiä välttävän polun etsiminen. Kuva 1 havainnollistaa asteroidien tutkimisen eri vaiheita avaruudessa.



Kuva 1. Asteroidien tutkimus

Tekoälymenetelmistä perehdytään evoluutioalgoritmeihin (geneettiset algoritmit ja diffrentiaalievoluutio), parviälyyn, meemiseen algoritmiin sekä sumeaan logiikkaan. Parviälystä tutkielmassa esillä ovat muurahaiskoloniaoptimointi (Ant Colony Optimization, ACO) ja hiukkasparvioptimointi (Particle Swarm Optimization, PSO).

Työ etenee johdannon kautta toisen luvun perinteisiin optimointitekniikoihin, jotka edellyttävät matemaattisesti raskasta laskentaa. Luvussa 3 keskitytään nykyaikaisiin optimointimenetelmiin ja luvussa 4 käsitellään uusimpia tekoälymentelmien ratkaisuja avaruustutkimuksen ongelmiin. Luvussa 5 on työn yhteenveto.

2. Perinteinen optimointi

Matemaattiset optimointimenetelmät ovat suosittuja matematiikan lisäksi mm. tietojenkäsittelytieteessä ja taloustieteessä. Optimoinnin tarkoituksena on kriteerien perusteella löytää käytössä olevista vaihtoehdoista paras ratkaisu. Yksinkertaisimmillaan tehtävänä voi olla löytää reaalifunktion pienin tai suurin arvo.

Optimointimenetelmät perustuvat klassiseen variaatiolaskentaan ja muodostavat sovelletun matematiikan keskeisen työkalun. Matemaattinen optimoinnin teoria tutkii funktioiden suurimpien ja pienimpien arvojen etsimistä rajoitusehtojen sitomien muuttujien joukosta [Silvennoinen, 2004]. Optimointiteoria sisältää differentiaalilaskennan ääriarvoteorian, mutta on huomattavasti sitä laajempi. Optimointisovellusten kannalta on oleellista muodostaa sellainen malli, jossa kyseinen ilmiö kuvataan matemaattisesti niin, että optimointimenetelmiä voidaan käyttää.

Optimointiongelma on matemaattinen malli kuvattavasta ilmiöstä [Silvennoinen, 2004]. Mallin muuttujia sitovat analyyttiset yhtälöt tai epäyhtälöt, joita kutsutaan rajoitusyhtälöiksi ja rajoitusepäyhtälöiksi. Niitä voidaan nimittää yhteisesti rajoitusehdoiksi tai rajoitteiksi. Kaikki muuttujien arvot, jotka toteuttavat rajoitusehdot, ovat mallin käypiä ratkaisuja. Ratkaisujen paremmuutta vertaillaan erilaisten kriteerien avulla. Kriteerinä voi olla esimerkiksi polttoaineen kulutuksen minimointi tai luotettavuuden maksimointi. Perusmallissa optimoidaan yhden kriteerin suhteen kerrallaan, jolloin maksimoitava tai minimoitava kriteeri on optimointitehtävän kohdefunktio. Monitavoitteisessa optimoinnissa kohdefunktio on vektori, jonka komponentteina ovat optimoitavat kriteerit.

Tunnettuja matemaattisen optimoinnin alueita ovat konveksi optimointi, joka sisältää mm. lineaarisen optimoinnin, kokonaislukuoptimoinnin, kvadraattisen optimoinnin, epälineaarinen optimoinnin, kombinatorisen optimoinnin, stokastisen optimoinnin, heuristisen optimoinnin, dynaamisen optimoinnin ja optimiohjausteorian.

Optimointiongelmat voidaan luokitella muuttujien ja rajoitefunktioiden tyyppien mukaan. Muuttujien tyyppien mukaan jaettuna mallit ovat jatkuvia, kokonaislukumalleja tai sekalukumalleja. Jatkuvan mallin muuttujat ovat reaalilukuja, kokonaislukumallin kokonaislukuja ja sekalukumallissa on sekä reaali- että kokonaislukuja. Diskreetistä optimoinnista puhutaan, kun muuttujat saavat vain tietyn diskreetin joukon arvoja. Nämä arvot voidaan kuitenkin aina muuntaa kokonaislukumalleiksi. Kombinatorinen optimointi liittyy läheisesti diskreettiin optimointiin. Jatkuvien muuttujien tehtävien ratkaiseminen on yleensä helpompaa ja laskennallisesti vähemmän vaativaa kuin kokonais- tai sekalukuongelmien ratkaiseminen [Silvennoinen, 2004]. Käytännössä kuitenkin useat mallit sisältävät ainakin osana kokonaislukumuuttujia, joita tarvitaan esimerkiksi loogisia ehtoja mallinnettaessa.

Optimointiongelmat voidaan luokitella myös niissä esiintyvien funktioiden tyyppien mukaan. Mallin kaikkien funktioiden ollessa muotoa lineaarinen funktio + vakio on kyseessä lineaarisen optimoinnin ongelma (Linear Programming eli LP). Lineaarisella optimoinnilla on erityisasema optimoinnissa, mm. sen takia, että sille on olemassa muihin luokkiin verrattuna ylivoimaisen tehokkaita

ratkaisualgoritmeja [Silvennoinen, 2004]. Jos mallissa on epälineaarisia yhtälöitä, on kyseessä epälineaarisen optimoinnin ongelma (Nonlinear Programming eli NLP). NLP voidaan edelleen jakaa osaluokkiin, kuten konveksi ja kvadraattinen optimointi.

LP ja NLP ovat äärellisuloitteisia ongelmia. Jos ongelmien muotoilussa tarvitaan ääretönuloitteisia avaruuksia, niin silloin ollaan tekemisissä variaatiolaskennan tai optimisäätöteorian kanssa, joiden käsittely edellyttää funktionaalianalyysin käyttöä [Silvennoinen, 2010].

Yhdistettäessä molemmat edellä mainitut luokittelut saadaan esimerkiksi sovellusten kannalta kattavimmat lineaarinen sekalukuoptimointi (Mixed Integer LinearProgramming, MILP) ja epälineaarinen sekalukuoptimointi (MINLP). Pelkästään kokonaislukuja sisältävä lineaarinen optimointi esiintyy myös nimellä Integer Programming (IP), ja jos nämä kokonaisluvut ovat erityisesti kaikki binäärilukuja, on kyseessä binäärioptimointi eli Binary Integer Programming (BIP) [Silvennoinen, 2004].

2.1. Lineaarinen optimointi

Optimointiongelma on lineaarinen, jos kohdefunktio on lineaarinen funktio ja rajoitusehdot ovat lineaarisia yhtälöitä tai epäyhtälöitä. Lineaarisen optimoinnin perustehtävät voidaan aina esittää niin sanotussa standardimuodossa, joita on kaksi: standardi yhtälöongelma ja standardi epäyhtälöongelma. Olkoot n ja m kokonaislukuja.

Standardi yhtälöongelma on muotoa

$$\begin{aligned} \max z = {} & c_1x_1 + \ldots + c_nx_n \\ & a_{11}x_1 + \ldots + a_{1n}x_n = b_1 \\ & \vdots \qquad\qquad \vdots \\ & a_{m1}x_1 + \ldots + a_{mn}x_n = b_m \\ & x_1, \ldots, x_n \geq 0. \end{aligned}$$

Optimoinnin suuntana voi olla myös minimointi ja ongelma voidaan esittää myös matriisien avulla:

$$\begin{aligned} & \max z = c^T x \\ & Ax = b \\ & x \geq 0, \end{aligned}$$

missä $x \in \mathbb{R}^n$, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ ja A on $(m \times n)$-matriisi. Ongelman käypä joukko on silloin $S = \{x \in \mathbb{R}^n \mid Ax = b, x \geq 0\}$.

Standardi epäyhtälöongelma on muotoa

$$\begin{aligned} \max z = {} & c_1x_1 + \ldots + c_nx_n \\ & a_{11}x_1 + \ldots + a_{1n}x_n \leq b_1 \\ & \vdots \qquad\qquad \vdots \end{aligned}$$

$a_{m1}x_1 + ... + a_{mn}x_n \leq b_m$

$x_1,..., x_n \geq 0.$

Tässäkin optimoinnin suuntana voi olla myös minimointi. Ongelma voidaan esittää myös matriisien avulla:

$\max z = c^T x$

$Ax \leq b$

$x \geq 0,$

missä $x \in \mathbb{R}^n$, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ ja A on $(m \times n)$-matriisi. Ongelman käypä joukko on silloin $S = \{x \in \mathbb{R}^n \mid Ax \leq b, x \geq 0\}$.

Kaikki lineaarisen optimoinnin tehtävät voidaan muuntaa kumpaan tahansa näistä standardimuodoista [Silvennoinen, 2010].

Maksimointitehtävä voidaan aina palauttaa minimointitehtäväksi, sillä $\max f(x) = - \min (-f(x))$. Merkintä $x \geq 0$ tarkoittaa, että jokainen vektorin x komponenteista on ei-negatiivinen. Merkinnällä $c^T x$ tarkoitetaan vektorien c ja x piste- eli sisätuloa avaruudessa $\mathbb{R}^n$ eli vektori $c \in \mathbb{R}^n$ tulkitaan $(n \times 1)$-matriisiksi.

Äärellisen monen lineaarisen epäyhtälön avulla määritelty joukko $\mathbb{R}^n$:ssä on konveksi monitahokas (polyhedron). Koska yhtälö on tulkittavissa kahdeksi epäyhtälöksi, on siis lineaarisen optimoinnin käypä joukko aina monitahokas. Tasossa $\mathbb{R}^2$ monitahokas on monikulmio [Silvennoinen, 2010].

Ensimmäisen metodin lineaarisen optimoinnin ratkaisemiseksi kehitti Leonid Kantorovich vuonna 1937. Tämän simplex-algoritmina tunnetun menetelmän julkaisi George Dantzig vuonna 1947, ja se on edelleen käytetyimpiä optimointialgoritmeja. Menetelmä perustuu siihen, että optimiratkaisuina on aina käyvän joukon kärkipiste tai sitten äärellistä optimiratkaisua ei ole. Kun LP-ongelmaa tarkastellaan graafisesti, niin huomataan ratkaisun sijaitsevan aina kulmapisteessä eli ääriarvopisteessä (extreme point). Simplex-menetelmässä ääriarvopisteitä, ja tehtävän ratkaisua etsitään algebrallisesti Gaussin eliminointia käyttäen. LP-tehtävän kohdefunktio on aina konveksi, joten tehtävälle löytyy globaali optimi hyödyntämällä konveksin optimoinnin tuloksia. Toinen samantyylinen (Basis-exchange pivoting algorithm) menetelmä on Criss-cross-algoritmi [Terlaky and Zhang, 1993].

Jokaiselle lineaarisen optimoinnin probleemalle voidaan määritellä duaaliprobleema, jossa maksimointi vaihtuu minimoinniksi ja päinvastoin. Kohdefunktion kerroinvektori ja oikea puoli vaihtavat paikkaa. Kerroinmatriisi transponoituu ja kutakin primaalin rajoitusehtoa vastaa duaalin muuttuja ja päinvastoin. Eli, jos primaali on maksimointitehtävä, niin duaalissa minimoidaan ja päinvastoin. Samoin primaalin rajoitusehtoa vastaa duaalin muuttuja ja rajoitusehdon tyyppi (yhtälö tai epäyhtälö) määrää duaalin muuttujan etumerkkiehdon. Vastaavasti primaalin muuttujan etumerkkiehto määrää duaalin rajoitusehdon tyypin. Näistä voidaan osoittaa, että duaalin duaali on primaali. Duaalitehtävän pääasiallinen käyttö on tarjota ala- tai ylärajoja lineaarisen optimoinnin probleeman kohdefunktiolle. Näin mikä hyvänsä duaalitehtävän käypä ratkaisu antaa alarajan minimointitehtävän kohdefunktiolle

ja mikä hyvänsä käypä primaalitehtävän ratkaisu antaa ylärajan maksimointitehtävän kohdefunktiolle. Tästä seuraa että, jos duaalitehtävän ratkaisu on max w = ∞, niin primaalin käypä joukko on tyhjä ja jos primaalin ratkaisu on min z = -∞, niin duaalin käypä joukko on tyhjä. [Silvennoinen, 2010]

Usein on mahdollista, että optimoitavia suureita on monta ja ne voivat olla keskenään ristiriitaisia sekä yhteismitattomia, jolloin yhden kohdefunktion optimaalisuuskäsitettä on laajennettava. Tästä saadaan monitavoitteisen lineaarisen optimoinnin ongelma

$$\min z = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix} = Cx,$$

ehdolla

$Ax = b$

$x \geq 0,$

kun z_1, z_2,…, z_m ovat kriteerejä eli vektoriarvoisen kohdefunktion komponentteja [Silvennoinen, 2010].

Käypä ratkaisu **x*** on monitavoitteisen lineaarisen optimoinnin Pareto-optimaalinen ratkaisu, jos seuraava ehto on voimassa kaikille käyville ratkaisuille **x**:

$Cx \leq Cx^* \Rightarrow x = x^*$.

Tämä merkitsee, että jos jokin $C\mathbf{x}$:n komponentti on aidosti pienempi kuin $C\mathbf{x}^*$:n vastaava komponentti, niin jonkin toisen komponentin on oltava vastaavasti aidosti huonompi. Pareto-optimaalista ratkaisua voidaan siis parantaa jonkin kriteerin suhteen vain, jos se tehdään jonkin toisen kriteerin kustannuksella. Useimmiten Pareto-optimien laskeminen palautetaan yksitavoitteisten tehtävien ratkaisemiseksi, jota kutsutaan skalaarisointimenetelmäksi [Silvennoinen, 2010].

Interior-point eli sisäpistemenetelmä on Basis exchange -menetelmien lisäksi toinen suosittu tutkimussuunta lineaarisessa optimoinnissa. Sisäpistemenetelmiin lukeutuvat ellipsoidi-menetelmä [Khachiyan, 1979], Karmarkarin projektiivinen menetelmä [Karmarkar, 1984] sekä 1990-luvulla kehitetty ”barrier function” tai ”path-following”-menetelmä [Gondzio and Terlaky, 1996].

Ceselli ja muut [2012] ovat kehittäneet lineaarisella optimointimenetelmällä branch-and-cut-and-price -algoritmin, jolla saadaan optimaalinen kulkureitti planeetan pinnalla tutkimustyötä tekevälle kulkuneuvolle.

2.2. Epälineaarinen optimointi

Jos optimointiongelmassa on mukana epälineaarisia funktioita, kyseessä on epälineaarisen optimoinnin ongelma (Nonlinear Programming, NLP). Tämä voidaan edelleen jakaa osaluokkiin, kuten konveksi ja kvadraattinen optimointi. Edellä mainitut ongelmat ovat äärellisuloitteisia. Jos ongelmien

muotoilussa tarvitaan ääretönulotteisia avaruuksia, päädytään variaatiolaskentaan tai optimisäätöteoriaan, joiden käsittely edellyttää funktionaalianalyysin käyttöä [Silvennoinen, 2010].

Epälineaarinen optimointitehtävä on muotoa

$\min f(\mathrm{x})$

$h_i(\mathrm{x}) = 0,\ i = 1,\ldots, r$

$g_j(\mathrm{x}) = 0, j = 1,\ldots, s$

$\mathrm{x} \in \mathbb{R}^n$.

Funktio f on skalaariarvoinen kohdefunktio. Funktiot g_j ja h_i rajoitefunktioita ja muuttujavektori x kuuluu muuttuja-avaruuteen $\mathbb{R}^n$. Joukko

$S = \{\mathrm{x} \in \mathbb{R}^n \mid h_i(\mathrm{x}, \mathrm{y}) = 0,\ i = 1,\ldots, r,\ g_j(\mathrm{x}, \mathrm{y}) \leq 0, j = 1,\ldots, s\}$

on käypä joukko, jonka pisteet ovat optimointitehtävän käyvät ratkaisut. Tehtävän optimiratkaisu on sellainen käypä ratkaisu, joka antaa kohdefunktiolle pienimmän arvon. Minimoinnin sijasta voidaan funktiota f maksimoida, tai käyttää yhteyttä $\max f(\mathrm{x}) = - \min(-f(\mathrm{x}))$.

2.2.1. Epälineaarinen optimointi ilman rajoitteita

Jos ainakin yksi optimointitehtävässä esiintyvistä funktioista on epälineaarinen, kyseessä on epälineaarisen optimoinnin (non-linear mathematical programming, non-linear optimization) tehtävä. Ongelma, jolla ei ole rajoitteita (rajoitusehtoja, constraints), on muotoa

$$\min_{\mathrm{x}\in\mathbb{R}^n} f(\mathrm{x}).$$

Optimoinnin kannalta on tärkeää määritellä funktion jatkuvuus ja differentioituvuus, lokaali minimi- ja maksimikohta, ensimmäisen ja toisen kertaluvun approksimaatio, välttämätön ensimmäisen ja toisen kertaluvun ehto lokaalille ääriarvolle, Hessen matriisi, kriittiset pisteet sekä riittävä ehto lokaalille minimille ja maksimille.

Reaaliarvoinen kohdefunktio f oletetaan määritellyksi koko avaruudessa $\mathbb{R}^n$, ja f on siellä ensimmäisen kertaluvun osittaisderivaattoineen jatkuva ja differentioituva.

Lokaali minimikohta tarkoittaa, että on olemassa sellainen pisteen x* avoin ympäristö $B_\varepsilon(\mathrm{x}^*)$, että

$f(\mathrm{x}^*) \leq f(\mathrm{x}),\ \forall \mathrm{x} \in B_\varepsilon(\mathrm{x}^*)$.

Lokaali maksimikohta määritellään vastaavasti.

Riittävän säännöllisellä funktiolla f (osittaisderivaatat jatkuvia) on voimassa lineaarinen eli ensimmäisen kertaluvun approksimaatio

$f(\mathrm{x}^* + \mathrm{h}) = f(\mathrm{x}^*) + \nabla(\mathrm{x}^*)^T\mathrm{h} + \varepsilon(\mathrm{h})\|\mathrm{h}\|$.

Välttämätön ensimmäisen kertaluvun ehto lokaalille ääriarvolle on seuraava:

Jos x on jatkuvasti differentioituvan funktion* $f: \mathbb{R}^n \rightarrow \mathbb{R}$ *lokaali minimikohta, niin*

$\nabla f(\mathrm{x}^*) = 0$.

Sama ehto saadaan myös lokaalille maksimikohdalle. Niiden erottamiseksi tarvitaan toisen kertaluvun derivaattoja.

Ensimmäistä kertalukua tarkempi approksimaatio saadaan toisen kertaluvun approksimaatiolla

$f(x^* + h) = f(x^*) + \nabla(x^*)^T h + \frac{1}{2} h^T f''(x^*) h + \varepsilon(h)\|h\|^2$.

missä $f''(x^*) = H_f(x^*)$ on funktion f toinen derivaatta eli ns. Hessen matriisi.

Jos funktiolla f on olemassa toisen kertaluvun osittaisderivaatat, niin niistä koostuva Hessen matriisi H_f on

$$H_f = \begin{bmatrix} D_{11}f & D_{12}f & \cdots & \cdots & D_{1n}f \\ D_{21}f & D_{22}f & \cdots & \cdots & D_{2n}f \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ D_{n1}f & D_{n2}f & \cdots & \cdots & D_{nn}f \end{bmatrix},$$

missä on merkitty

$D_{ij}f = \frac{\partial^2 f}{\partial x_i \partial x_j}$.

Tästä eteenpäin oletetaan, että funktion f kaikki toisenkin kertaluvun derivaatat ovat jatkuvia. Silloin sekaderivaatat voidaan laskea missä järjestyksessä hyvänsä, joten $D_{ij}f = D_{ji}f$ eli Hessen matriisi on symmetrinen [Silvennoinen, 2010].

Välttämätön toisen kertaluvun ehto lokaalille ääriarvolle:

Olkoot funktio $f: \mathbb{R}^n \to \mathbb{R}$ *ja sen osittaisderivaatat toiseen kertalukuun asti jatkuvia. Jos* x^* *on funktion f lokaali minimikohta, niin funktion f gradientti kohdassa* x^* *häviää ja Hessen matriisi on siinä positiivisesti semidefiniitti:*

$\nabla f(x^*) = 0$ *ja* $H_f(x^*) \geq 0$.

Jos x^* *on f:n lokaali maksimikohta, niin funktion f gradientti kohdassa* x^* *häviää ja Hessen matriisi on siinä negatiivisesti semidefiniitti:*

$\nabla f(x^*) = 0$ *ja* $H_f(x^*) \leq 0$.

Derivaattamerkinnöillä ehto saa muodon:

$f'(x^*) = 0$ ja $f''(x^*) \geq 0$ lokaalissa minimikohdassa x^*

$f'(x^*) = 0$ ja $f''(x^*) \leq 0$ lokaalissa maksimikohdassa x^*.

Funktion f kriittisiksi pisteiksi sanotaan kaikkia niitä pisteitä x, joissa funktion gradientti on nolla. Joskus myös mahdolliset funktion tai sen osittaisderivaattojen epäjatkuvuuskohdat otetaan mukaan kriittisiin pisteisiin, vaikka niissä eivät ääriarvoehdot ole voimassa. Ne kriittiset pisteet, joissa gradientti on nolla, mutta jotka eivät ole lokaaleja minimejä tai maksimeja, ovat satulapisteitä. Kriittisten pisteiden "laadun" tutkimiseksi (eli ovatko lokaaleja minimejä, maksimeja jne.) voidaan käyttää riittäviä ehtoja. Näistä tunnetuin on yhden muuttujan funktioiden ehdon

$f''(x) = 0 \;\&\; f''(x) > 0 \Rightarrow x$ lokaali minimikohta.

Riittävä ehto lokaalille minimille ja maksimille on seuraava:

Olkoot funktio $f: \mathbb{R}^n \to \mathbb{R}$ *ja sen osittaisderivaatat toiseen kertalukuun asti jatkuvia sekä* x^* *kriittinen piste eli*

$\nabla f(x^*) = 0$.

Jos lisäksi f:n Hessen matriisi $H_f(x^*)$ *on positiivisesti definiitti, niin* x^* *on lokaali minimikohta, ja jos negatiivisesti definiitti, niin* x^* *on lokaali maksimikohta, niin*

$f''(x^*) > 0 \Rightarrow x^*$ *lokaali minimikohta*

$f''(x^*) < 0 \Rightarrow x^*$ *lokaali maksimikohta.*

Jos Hessen matriisi $H_f(x^*)$ *on indefiniitti, niin* x^* *on satulapiste.*

2.2.2. Optimaalisuusehtoja, epäyhtälö- ja yhtälörajoitukset

Tarkastellaan yleisesti muotoiltua ongelmaa

$$\min_{x \in S} f(x),$$

missä $S \subset \mathbb{R}^n$. Tällä ongelmalla on seuraava yleinen optimaalisuusehto:

Jos x^* on funktion f lokaali minimikohta joukossa S, niin

$D^<_f(x^*) \cap K_S(x^*) = \emptyset$, missä

$D^<_f(x)$ = funktion f vähenemissuuntien kartio pisteessä x,

$K_S(x)$ = joukon S käypien suuntien kartio pisteessä x,

$T_S(x)$ = joukon S tangenttisuuntien kartio pisteessä x.

Koska tapauksessa $S = \mathbb{R}^n$ on $K_S(x^*) = \mathbb{R}^n \setminus \{0\}$, saadaan tästä myös ehdot vapaille optimointitehtäville. Ne pätevät myös tapauksiin, joissa piste x^* on joukon S sisäpiste.

Fritz Johnin ehdot ovat voimassa sellaisessa tapauksessa, jossa x^* on käyvän joukon reunapiste ja ongelman rajoituksena on vain epäyhtälöitä:

Jos x^* on tehtävän

$$\min f(x)$$
$$g_i(x) \leq 0,\ i = 1,\ldots, m$$

lokaali minimikohta, niin on olemassa sellaiset ei-negatiiviset kertoimet $\mu_0, \mu_1, \ldots, \mu_m$, joista ainakin yksi on $\neq 0$, että

$$\mu_0 \nabla f(x^*) + \mu_1 \nabla g_1(x^*) + \ldots + \mu_m \nabla g_m(x^*) = 0 \qquad (1)$$

$$\mu_i g_i(x^*) = 0,\ i = 1,\ldots, m \qquad (2)$$

Kertoimia μ_i kutsutaan Lagrangen kertoimiksi. Ehdot (2) merkitsevät, että jos rajoitus i on epäaktiivinen eli $g_i(x^*) < 0$, niin vastaava kerroin $\mu_i = 0$.

Fritz Johnin ehdot takaavat, että kertoimista μ_i ainakin yksi on nollasta poikkeava. Voi siis käydä, että $\mu_0 = 0$, jolloin ehto (1) ei sano mitään kohdefunktiosta. Tietyin lisäehdoin voidaan kuitenkin taata, että $\mu_0 > 0$ ja tällöin ehtoja (1)-(2) sanotaan Karushin-Kuhnin-Tuckerin ehdoiksi, lyhennyksenä KKT. [Silvennoinen, 2010]

Jos optimointitehtävän rajoitusehdoissa on epälineaarisia yhtälöitä, niin käypien suuntien kartio $K_S(x)$ on yleensä tyhjä jokaisessa pisteessä $x \in S$. Tällöin yleinen optimaalisuusehto, $D^<_f(x^*) \cap K_S(x^*) = \emptyset$, ei kerro mitään ja on otettava käyttöön tangenttikartion käsite, jolloin vastaava optimaalisuusehto saa muodon:

Jos x^* on differentioituvan funktion f lokaali minimikohta joukossa S, niin

$\{d \mid \nabla f(x^*)^T d < 0\} \cap T_S(x^*) = \emptyset$.

Tarkastelemalla optimointitehtävää, jossa käypä joukko on määritelty sekä epäyhtälö- että yhtälörajoituksilla saadaan ehdot muotoa

$$\min_{x\in\mathbb{R}^n} f(x)$$
$$g_i(x) \leq 0,\ i = 1,\ldots, m$$
$$h_j(x) = 0, j = 1,\ldots, s. \qquad (3)$$

Tässä oletetaan, että kaikki esiintyvät funktiot ovat differentioituvia. Rajoitusehtojen laatuvaatimus (constraint qualification) on voimassa pisteessä x^0, jos

$$T_S(x^0) = \{ d\in\mathbb{R}^n \mid \nabla g_i(x^0)^T d \leq 0,\ i = 1,\ldots, m\} \cap \{ d\in\mathbb{R}^n \mid \nabla h_j(x^0)^T d = 0, j = 1,\ldots, s\}.$$

Tällöin saamme välttämättömäksi optimaalisuusehdoksi (Karush-Kuhn-Tucker):

Jos x* on ongelman (3) lokaali minimikohta ja rajoitusehtojen laatuvaatimus on voimassa pisteessä x*, niin on olemassa sellaiset kertoimet $\mu_i \geq 0$, (i =1,…, m) ja λ_j (j =1,…, s), että

$$\nabla f(x^*) + \mu_1\nabla g_1(x^*) +\ldots+ \mu_m\nabla g_m(x^*) + \lambda_1\nabla h_1(x^*) +\ldots+ \lambda_s\nabla h_s(x^*) = 0$$
$$\mu_1\nabla g_1(x^*) = 0,\ i = 1,\ldots, m.$$

2.2.3. Konveksin optimoinnin ehdoista

Minimointitehtäviä, joissa kohdefunktio on konveksi funktio ja käypä joukko konveksi joukko, sanotaan konvekseiksi ongelmiksi. Sellaisissa kaikki lokaalit minimit ovat globaaleja. Konveksien optimointitehtävien optimaalisuusehdot ovat usein samalla kertaa välttämättömiä ja riittäviä. Seuraava ehto ei edellytä edes differentioituvuutta kohdefunktiolle:

Piste x* on konveksin funktion f minimikohta konveksissa joukossa S, jos ja vain jos pisteessä x* on olemassa funktion f subgradientti k siten, että

$$k^T(x - x^*) \geq 0,\ \forall x \in S.$$

Tästä seuraa sisäpisteiden tapauksessa ehdon $\nabla f(x^*) = 0$ yleistys ei-differentioituville konvekseille funktioille:

Konveksin joukon S sisäpiste x* on konveksin funktion f minimikohta joukossa S, jos ja vain jos 0 kuuluu f:n subdifferentiaaliin eli $0 \in \partial f(x^*)$.

KKT-ehdot ovat tietyin konveksisuusehdoin myös riittäviä:

Olkoon x* ongelman

$$\min_{x\in\mathbb{R}^n} f(x)$$
$$g_i(x) \leq 0,\ i = 1,\ldots, m$$
$$h_j(x) = 0, j = 1,\ldots, s$$

käypä ratkaisu ja $J(x^*) = \{i \mid g_i(x^*) = 0\}$. Oletetaan, että KKT-ehdot ovat pisteessä x* voimassa eli että funktiot ovat differentioituvia ja on olemassa sellaiset kertoimet $\mu_i \geq 0$, (i = 1,…, m) ja λ_j (j = 1,…, s), että

$$\nabla f(x^*) + \sum_{i\in J(x^*)} \mu_i\nabla g_i(x^*) + \sum_{j=1}^{s} \lambda_j\nabla h_j(x^*) = 0.$$

Jos funktiot f, g_i, $i \in J(x^*)$ ja h_j, $j \in J^+ = \{j \mid \lambda_j > 0\}$ ovat konvekseja ja funktiot h_j, $j \in J^- = \{j \mid \lambda_j < 0\}$ konkaaveja, niin x* on ongelman globaali minimiratkaisu.

Edellä esitetyt oletukset ovat voimassa esimerkiksi, jos kohdefunktio on konveksi ja tehtävässä on pelkästään epäyhtälörajoituksia $g_i(\mathrm{x}^*) \leq 0$, missä funktiot g_i ovat konvekseja. Silloin käypä joukko on konveksi, joten kyseessä on konveksi ongelma. Jos rajoitusehtojen laatuvaatimus on voimassa kaikissa käyvissä pisteissä, ovat Karushin-Kuhnin-Tuckerin ehdot silloin välttämätön ja riittävä ehto optimaalisuudelle. [Silvennoinen, 2010]

2.2.4. Yksiulotteinen optimointi

Tässä alakohdassa tarkastellaan yhden reaalimuuttujan optimointitehtävien ratkaisumenetelmiä eli muotoa

$$\min_{\mathrm{x} \in I} f(\mathrm{x})$$

olevia ongelmia. Tällaiset tehtävät ovat monen n-ulotteisen optimointitehtävän aliongelmia, joissa haetaan suurinta tai pienintä arvoa puolisuoralta. Väli I on joko kompakti tai suljettu väli $[a, \infty)$ tai koko $\mathbb{R}$. Seuraavassa esiteltävät menetelmät ovat klassisia perusalgoritmeja, jotka eivät välttämättä ole tehokkaimpia mahdollisia, mutta jotka monesti riittävät ja ovat yksinkertaisia ohjelmoida. Menetelmät voidaan jakaa derivaattaa käyttäviin ja käyttämättömiin. Voidaan myös eritellä simultaanihaut, jonomenetelmät ja approksimointimenetelmät. [Silvennoinen, 2010]

Hilamenettely (Tasainen haku, Uniform Search)

Jaetaan kompakti väli $I = [a, b]$ pisteillä $x_0 = a, x_1, \ldots, x_{n-1}, x_n = b$ yleensä yhtä pitkiin osaväleihin, joiden pituus on halutun tarkkuuden suuruusluokasta puolet. Määritetään

$$f(\mathrm{x}^*) = \min\{f(x_i) \mid i = 0,1,\ldots, n\}$$

ja uusitaan menettely tarvittaessa pisteen x* ympäristössä. [Silvennoinen, 2010]

Satunnaishaku

Muuten samanlainen kuin hilamenettely, mutta nyt välille $I = [a, b]$ muodostetaan jakopisteet satunnaisesti satunnaislukugeneraattorilla. Yleensä käytetään tasaisesti jakautuneita (pseudo)satunnaislukuja, mutta myös kvasisatunnaislukuja on alettu käyttää enenevässä määrin (ne eivät jätä niin suuria aukkoja välille I kuin pseudosatunnaisluvut oikeaoppisesti voivat jättää). [Silvennoinen, 2010]

Rosenbrockin menetelmä

Olkoon $I = \mathbb{R}$. Valitaan alkupiste a, askepituus h ja haluttu tarkkuus δ. Lasketaan $f(a)$. Siirrytään pisteeseen $a + h$ ja lasketaan $f(a + h)$. Jos $f(a + h) < f(a)$, jatketaan samaan suuntaan, mutta kaksinkertaisella askelpituudella, eli siirrytään pisteeseen $(a + h) + 2h = a + 3h$. Näin jatketaan, kunnes funktio alkaa kasvaa, jolloin vaihdetaan suuntaa ja mennään taaksepäin neljäsosalla edellisestä askelpituudesta. [Silvennoinen, 2010]

Väärän position menetelmä (Method of False Position)

Kahdesta peräkkäisestä iteraatiopisteestä x_{k-1}, x_k lasketaan uusi. Oletetaan, että tunnetaan $f'(x_{k-1})$, $f(x_k)$, $f'(x_k)$. Haetaan sellainen toisen asteen polynomi p, että $p'(x_{k-1}) = f'(x_{k-1})$, $p(x_k) = f(x_k)$, $p'(x_k) = f'(x_k)$.

Silloin

$$p(x) = f(x_k) + f'(x_k)(x-x_k) + (f'(x_{k-1}) - f'(x_k))/(x_{k-1} - x_k)*½(x-x_k)^2,$$

josta optimikohdalle saadaan

$$p'(x_{k+1}) = f'(x_k) + (f'(x_{k-1}) - f'(x_k))/(x_{k-1} - x_k)*(x_{k+1} - x_k) = 0.$$

Saadaan siis

$$x_{k+1} = x_k - f'(x_k)*(x_{k-1} - x_k)/(f'(x_{k-1}) - f'(x_k)).$$

Epälineaarisen optimoinnin ratkaisualgoritmit eivät yleensä ole tarkkoja, sillä ne eivät konvergoi optimiin äärellisen monen askeleen päästä, kuten lineaarisessa optimoinnissa. [Silvennoinen, 2010]

2.2.5. Gradienttimenetelmä

Tarkastellaan ongelmaa

$$\min_{x \in \mathbb{R}^n} f(x),$$

missä f on jatkuvasti differentioituva. Koska funktio pienenee voimakkaimmin gradientin vastasuuntaan mentäessä, voidaan iterointisuuntaa valittaessa ottaa se etenemissuunnaksi. Tällainen "ahne" menettely ei aina ole paras, mutta tarjoaa hyvän pohjan erilaisille algoritmi-ideoille. Aloitetaan jostakin pisteestä x_0. Uusi iteraatiopiste on

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k), \qquad (4)$$

missä

$$\alpha_k = \underset{\alpha \geq 0}{\operatorname{argmin}} f(x_k - \alpha \nabla f(x_k)). \qquad (5)$$

Menetelmästä käytetään nimitystä gradienttimenetelmä eli jyrkimmän laskun menetelmä (method of steepest descent, SD). Askelpituuden α_k määrittäminen yhtälössä (5) voi tapahtua periaatteessa tarkasti, tai siihen voidaan käyttää jotakin heuristista sääntöä. Tässä oletetaan, että α_k on tarkka minimikohta ehdon (5) määrittelemässä haussa puolisuoralta. SD-menetelmällä on se hyvä puoli, että jos algoritmi ei pääty, niin jokaisessa askeleessa saavutetaan funktiolle f pienempi arvo. Seuraavassa alakohdassa esiteltävällä Newtonin menetelmällä ei välttämättä ole tätä ominaisuutta. Toinen SD-menetelmän hyvä ominaisuus on, että sillä on vahvat konvergenssiominaisuudet. [Laitinen, 2014]

Voidaan todistaa, että jos

$$f \in C^1(\mathbb{R}^n) \text{ ja } \lim_{|x| \to \infty} f(x) = \infty,$$

niin yhtälöiden (4) ja (5) määräämä menetelmä konvergoi kohti lokaalia minimiä.

Gradienttimenetelmä saa yksinkertaisen muodon kvadraattisille funktioille:

$$f(x) = ½x^T Q x - b^T x,$$

missä Q on symmetrinen ja positiivisesti definiitti. Silloin

$$\nabla f(x) = Qx - b, \; H_f(x) = Q.$$

Gradienttimenetelmä saa nyt muodon

$x_{k+1} = x_k - \alpha_k g_k$, $g_k = Qx_k - b$

$\alpha_k = \underset{\alpha \geq 0}{\operatorname{argmin}} f(x_k - \alpha g_k) = g^T_k g_k / g^T_k Q g_k$.

Tässä kertoimen α_k lauseke on saatu derivoimalla $f(x_k - \alpha g_k)$ muuttujan α suhteen ja ratkaisemalla 1. kertaluvun ääriarvoehto.

Jos merkitään x*:llä funktion *f* globaalia optimikohtaa ja

$E(x) = \frac{1}{2}(x-x^*)^T Q(x-x^*)$,

voidaan osoittaa gradienttimenetelmän suppenemiselle ehto

$E(x_{k+1}) \leq ((A-a)/(A+a))^2 E(x_k)$,

missä *A* on *Q*:n suurin ja *a* pienin ominaisarvo.

Siis jono $(E(x_k))$ suppenee kohti nollaa lineaarisesti, suppenemissuhteena

$((A-a)/(A+a))^2$.

2.2.6. Newtonin menetelmiä

Tunnetuimpia derivaattoja käyttäviä approksimaatiomenetelmiä on Newtonin menetelmä. Newtonin menetelmässä approksimoidaan funktiota *f* kvadraattisella funktiolla, joka sitten minimoidaan. Käytetään pisteessä x_k laskettua toisen asteen Taylorin polynomia

$m_k(x) = \nabla f(x_k) + \nabla f(x_k)^T (x-x_k) + \frac{1}{2}(x-x_k)^T H_f(x_k)(x-x_k) \approx f(x)$.

Jos x_{k+1} on m_k:n minimikohta, $\nabla m_k(x_{k+1}) = 0$, niin saadaan

$\nabla f(x_{k+1}) + H_f(x_k)(x_{k+1}-x_k) = 0$,

josta saadaan Hessen matriisin ollessa säännöllinen

$x_{k+1} = x_k - H_f(x_k)^{-1} \nabla f(x_k)$.

Menetelmä ei ole globaalisti suppeneva. Jos *f* on kahdesti jatkuvasti differentioituva ja Hessen matriisi positiivisesti definiitti sekä Lipschitz-jatkuva, niin voidaan osoittaa Newtonin menetelmän konvergoivan riittävän läheltä lokaalia optimia, kun aloitetaan kvadraattisesti.

Jos $H_f(x_k)$ ei ole positiivisesti definiitti, suunta $d = -H_f(x_k)^{-1} \nabla f(x_k)$ ei välttämättä ole vähenemissuunta. Tilanne voidaan yrittää korjata modifioimalla hakusuuntaa Hessen matriisia muuttamalla. Valitaan

$G(x_k) = H_f(x_k) + \mu_k I$,

missä parametri μ_k on valittu sellaiseksi, että *G* on positiivisesti definiitti. Silloin

$x_{k+1} = x_k - G(x_k)^{-1} \nabla f(x_k)$

on modifioitu Newtonin menetelmä.

Kvasi-Newton menetelmissä ideana on korvata Hessen matriisi sellaisella approksimaatiolla, joka on Hessen matriisia helpompi päivittää iteraatiosta toiseen. Tällöin menetelmän kulku on yleisesti seuraava: Valitaan iteraation alkupiste x_0, lasketaan Hessen matriisin approksimaatio H_0. Iteraatiopisteessä x_k määritetään hakusuunta yhtälöstä

$H_k d_k = -\nabla f(x_k)$, $k = 0,1,2,...$

Askelpituus on

$t_k = \underset{t>0}{\operatorname{argmin}} f(\mathrm{x}_k + t\mathrm{d}_k)$.

Iterointi lopetetaan, kun lopetusehto on saavutettu. Muutoin Hessen matriisi päivitetään matriisiksi H_{k+1}. Erilaiset kvasi-Newton menetelmät eroavat toisistaan päivitystavan suhteen.

Yhtälöä

$H_k(\mathrm{x}_k) = \nabla f(\mathrm{x}_k) - \nabla f(\mathrm{x}_{k-1})$

sanotaan sekanttiyhtälöksi tai kvasi-Newton-yhtälöksi. Ottamalla käyttöön kirjallisuudessa laajalti esiintyvät merkinnät

$\mathrm{s}_k = \mathrm{x}_{k+1} - \mathrm{x}_k$, $\mathrm{y}_k = \nabla f(\mathrm{x}_{k+1}) - \nabla f(\mathrm{x}_k)$

sekanttiyhtälö saa muodon

$H_{k+1}\mathrm{s}_k = \mathrm{y}_k$.

Tämä voidaan esittää käänteismatriisin

$B_{k+1} = H^{-1}{}_{k+1}$

avulla:

$B_{k+1}\mathrm{y}_k = \mathrm{s}_k$.

Yksinkertaisimpia päivityksiä on Broydenin astetta 1 oleva päivitys:

$B_{k+1} = B_k + \alpha \mathrm{u}\mathrm{u}^T$.

Sijoittamalla tämä sekanttiyhtälöön saadaan

$\mathrm{s}_k = B_{k+1}\mathrm{y}_k = B_k\mathrm{y}_k + \alpha \mathrm{u}\mathrm{u}^T\mathrm{y}_k$,

josta assosiatiivisuuden ja sisätulon kommutatiivisuuden avulla pätee

$\alpha(\mathrm{u}^T\mathrm{y}_k)\mathrm{u} = \mathrm{s}_k - B_k\mathrm{y}_k$.

Jos valitaan $\mathrm{u} = \mathrm{s}_k - B_k\mathrm{y}_k$, nähdään, että yhtälö toteutuu, kun

$\alpha = ((\mathrm{s}_k - B_k\mathrm{y}_k)^T\mathrm{y}_k)^{-1}$,

josta seuraa päivitykselle lauseke

$B_{k+1} = B_k + ((\mathrm{s}_k - B_k\mathrm{y}_k)(\mathrm{s}_k - B_k\mathrm{y}_k)^T)/((\mathrm{s}_k - B_k\mathrm{y}_k)^T\mathrm{y}_k)$.

Nähdään, että symmetrisyys säilyy ja $\mathrm{rank}(B_{k+1} - B_k) = 1$. Positiividefiniittisyys ei kuitenkaan välttämättä säily.

Toinen, monimutkaisempi päivitysmenetelmä on BFGS (Broyden-Fletcher-Goldfarb-Shannon):

$\mathrm{x}^{(k+1)} = \mathrm{x}^{(k)} - t_k D_k^{-1}\nabla f(\mathrm{x}^{(k)})$.

Tämän menetelmän puutteena on se, että

$-D_k^{-1}\nabla f(\mathrm{x}^{(k)})$

ei välttämättä ole funktion f vähenemissuunta, koska D_k ei välttämättä ole positiivisesti definiitti. [Silvennoinen, 2010]

DFP-menetelmä (Davidon-Fletcher-Powell) on puolestaan BFGS-menetelmää varhaisempi. Tässä menetelmässä iteraatio esitetään muodossa

$\mathrm{x}^{(k+1)} = \mathrm{x}^{(k)} - t_k D_k \nabla f(\mathrm{x}^{(k)})$,

missä matriisi D_k on BFGS-menetelmän iteraatiomatriisin käänteismatriisi.

Selvä DFP-menetelmän etu verrattuna BFGS-menetelmään on, että uusi suunta saadaan siinä laskettua suoraan $p_k = D_k\nabla f(x^{(k)})$, kun taas BFGS-menetelmässä täytyy ratkaista yhtälöryhmä $D_k p_k =$

$\nabla f(x^{(k)})$. Vaikka molempien menetelmien iterointimatriisit ovat positiivisesti definiittejä, niin DFP-menetelmän matriisilla D_k on taipumus muodostua ei-positiividefiniitiksi tietokoneen pyöristysvirheiden vuoksi. [Laitinen, 2014]

Xinsheng ja muut [2006] tutkivat yhden avaruusaluksen liikkeen optimoimista kehittämällään kvasi-Newton menetelmällä. Tutkimuksessa ”cost”-funktionaalin minimointi formuloitiin epälineaarisen optimoinnin ongelmaksi, joka sitten ratkaistiin kvasi-Newton metodilla.

2.2.7. Konjugaattigradienttimenetelmä

Gradienttimenetelmän sukulainen konjugaattigradienttimenetelmä hakee uudet hakusuunnat kaikkien edellisten hakusuuntien kanssa kohtisuorista suunnista, ei ainoastaan edellisen. Ortogonaalisuuskäsite voidaan yleistää positiivisesti definiitin matriisin Q suhteen seuraavasti:

Vektorit x ja y ovat Q-ortogonaaliset eli ortogonaaliset Q:n suhteen, jos

$x^T Q y = 0$.

Jos $Q = I$, saadaan tavallinen ortogonaalisuus.

Vektorit $d_0, d_1, \ldots, d_k$ muodostavat Q-ortogonaalisen joukon, jos

$d_i^T Q d_j = 0, \forall\ i \neq j$.

Jos Q on positiivisesti definiitti, niin jokainen nollasta eroavien vektorien muodostama Q-ortogonaalinen joukko on lineaarisesti riippumaton.

Konjugaattigradienttialgoritmi kvadraattiselle funktiolle toimii seuraavasti:

Valitaan alkupiste $x_0 \in \mathbb{R}^n$,

$d_0 = -g_0 = b - Qx_0$,

$x_{k+1} = x_k + \alpha_k d_k$,

$\alpha_k = -(g_k^T d_k / d_k^T Q d_k)$,

$d_{k+1} = -g_{k+1} + \beta_k d_k$,

$\beta_k = (g^T_{k+1} Q d_k / d_k^T Q d_k)$,

$g_k = Qx_k - b$.

Verrattuna gradienttimenetelmään iteraation askelten lukumäärä on siis äärellinen n. Menetelmää käytetään varsin paljon suurten lineaaristen yhtälöiden ratkaisemiseen. Menetelmän yleistäminen ei-kvadraattisille funktioille voi tapahtua usealla tavalla. Eräs mahdollisuus on approksimoida funktiota kvadraattisella polynomilla, jolloin konjugaattigradienttialgoritmissa tehdään korvaukset

$g_k \leftrightarrow \nabla f(x_k)$, $Q \leftrightarrow H_f(x_k)$.

Silloin alkupisteestä x_0 lähtien iteroidaan algoritmia n kertaa, asetetaan $x_0 = x_n$ ja toistetaan. Algoritmin etuna on, että puolisuoralta hakua ei tarvita. Haittana taas on se, että Hessen matriisi on laskettava joka iterointikierroksella. [Silvennoinen, 2010]

2.2.8. Sakkofunktiomenetelmät

Sakkofunktiomenetelmällä voidaan ratkaista epäyhtälöllä rajoitettu optimointitehtävä käyttämällä apuna rajoittamattoman optimoinnin tehtävää. Tämä aputehtävä muodostetaan alkuperäisen tehtävän kohdefunktion ja rajoitteiden avulla. Aputehtävä sisältää sakkoparametrin, joten itse asiassa käytetään sopivaa jonoa aputehtäviä. [Laitinen, 2014]

Rajoitusehdoilla varustetun tehtävän yleinen muoto on

$$\min_{x \in S} f(x). \qquad (6)$$

Sakkofunktioiden idea on seuraava: Approksimoidaan tehtävää (6) vapailla optimointiongelmilla, joissa on muunnettu kohdefunktio

$$f(x) + G(x).$$

Näissä G on sakkofunktio, joka sakottaa mentäessä käyvän joukon S ulkopuolelle. [Silvennoinen, 2010]

Ulkoinen sakkofunktiomenetelmä

Korvataan ongelma (6) vapaalla ongelmalla

$$\min_{x \in \mathbb{R}^n} f(x) + \mu P(x), \qquad (7)$$

missä parametri $\mu > 0$ on vakio, funktio P: $\mathbb{R}^n \rightarrow \mathbb{R}$ on jatkuva ja ei-negatiivinen, ja

$$P(x) = 0 \Leftrightarrow x \in S.$$

Kun $\mu \rightarrow \infty$, ongelman (7) optimiratkaisu lähestyy ongelman (6) optimiratkaisua. Yleensä valitaan jono (μ_k), missä $\mu_k \rightarrow \infty$, ja aloituspiste x_0, jonka ei tarvitse olla käypä. Vapaan tehtävän (7) optimiratkaisut ovat silloin iterointipisteitä x_k. Jos käypä joukko on määritelty epäyhtälöillä

$$S = \{x \mid g_i(x) \leq 0,\ i = 1,\ldots, m\},$$

niin sakkofunktioksi valitaan joskus

$$P(x) = \sum_{i=1}^{m} (\max(0, g_i(x)))^2.$$

Tämä ei kuitenkaan ole differentioituva. Toinen tapa on muuttaa ensin epäyhtälöt yhtälöiksi

$$S = \{x \mid g_i(x) + u_i^2 = 0,\ i = 1,\ldots, m\},$$

jolloin sakkofunktioksi käy differentioituva funktio

$$P(x) = \sum_{i=1}^{m} (g_i(x) + u_i^2))^2.$$

Jos alkuperäisen tehtävän rajoitusehdot ovat yhtälömuotoiset:

$$S = \{x \mid h_i(x) = 0,\ i = 1,\ldots, m\},$$

niin silloin luonteva sakkofunktiovalinta on

$$P(x) = \sum_{i=1}^{m} h_i(x)^2.$$

Jos sakkofunktiomenetelmällä haetaan ongelman KKT-pistettä, puhutaan täydennetyn Lagrangefunktion menetelmästä. Yhtälörajoitteisen ongelman

$$\min f(x)$$
$$h(x) = 0 \qquad (8)$$

KKT-piste löydetään Lagrangen funktion

$$L(x,\lambda) = f(x) + \lambda^T h(x)$$

gradientin nollakohdista. Koska ongelman (8) käyvissä pisteissä on

$L(\mathrm{x},\lambda) = f(\mathrm{x})$,

on ongelma (8) yhtäpitävä ongelman

$\min L(\mathrm{x},\lambda)$

$\mathrm{h}(\mathrm{x}) = 0$

kanssa. Kun tähän sovelletaan sakkofunktiomenetelmää, saadaan minimoitavaksi täydennetty Lagrangen funktio (augmented Lagrangian)

$A(\mathrm{x}, \lambda, \mu) = f(\mathrm{x}) + \lambda^T \mathrm{h}(\mathrm{x}) + \frac{\mu}{2}\mathrm{h}(\mathrm{x})^T\mathrm{h}(\mathrm{x})$. [Silvennoinen, 2010]

Sisäinen sakkofunktiomenetelmä

Toinen sakotuskäytäntö on estää käyvästä joukosta poistuminen, jos siellä alkupisteenä jo ollaan. Tällöin poistumisen estää sisäinen sakkofunktio eli estefunktio. Ongelma (6) korvataan vapaalla ongelmalla

$\min f(\mathrm{x}) + \frac{1}{\mu k}B(\mathrm{x})$,

missä parametrijono $\mu_k > 0$, funktio B on jatkuva ja $B(\mathrm{x}) \to \infty$, kun $\mathrm{x} \to \partial S$ (∂S on joukon S reuna). Nyt aloituspisteen x_0 on oltava käypä. Sisäinen sakkofunktiomenetelmä soveltuu vain tehtäviin, joissa käyvällä joukolla on sisäpisteitä. Siis vain epäyhtälörajoitukset ovat mahdollisia. Olkoon käypä joukko muotoa

$\mathrm{S} = \{\mathrm{x} \mid g_i(\mathrm{x}) \leq 0,\ i = 1,\ldots, m\}$.

Silloin käytettyjä estefunktioita ovat logaritmieste

$B(\mathrm{x}) = -\sum_{i=1}^{m} \ln(-g_i(\mathrm{x}))$

tai inverssieste

$B(\mathrm{x}) = -\sum_{i=1}^{m} \frac{1}{g_i(\mathrm{x})}$.

Jos tehtävässä on sekä yhtälö- että epäyhtälörajoituksia, voidaan ulkoista ja sisäistä sakkofunktiomenetelmää käyttää yhdessä, ulkoista yhtälörajoituksiin ja sisäistä epäyhtälörajoituksiin.

2.2.9. Leikkaustasomenetelmät

Leikkaustasomenetelmät (cutting plane) perustuvat seuraavaan ideaan: approksimoidaan konveksin ongelman

$$\min_{\mathrm{x}\in K} f(\mathrm{x}). \qquad (9)$$

käypää konveksia joukkoa K "ylhäältä päin" sitä laajemmalla konveksilla monitahokkaalla. Tätä sitten leikataan hypertasoilla pienemmäksi niin kauan, kunnes käypä optimipiste on löydetty. Menetelmän muotoilu on ongelmalle

$$\min_{\mathrm{x}\in M} \mathrm{c}^T\mathrm{x} \qquad (10)$$

missä kohdefunktio on lineaarinen ja M on suljettu konveksi joukko. Tämä muotoilu ei ole yleisyyttä rajoittava, koska tehtävä (9), jossa f on konveksi, voidaan muuntaa muotoon (10) seuraavasti:

$$\min_{\mathrm{x}\in K, s} s$$

$f(x) - s \leq 0$.

Siis monitahokkaita pienennetään leikkaamalla aina edellinen laajemman käyvän joukon optimikohta pois. Tällä tavalla lähestytään joukkoa M ja $x_k \rightarrow x^* \in M$, joka on ongelman (10) optimiratkaisu.

Eri algoritmit eroavat toisistaan siinä, miten hypertaso H_k valitaan. Käytetyin ratkaisu on Kelleyn leikkaustasomenetelmä.

Siinä optimointiongelma on muotoa

$\min c^T x$

$g_i(x) \leq 0,\ i = 1,\ldots, m,$

missä funktiot g_i ovat konvekseja ja differentioituvia. Silloin käypä joukko M on konveksi joukko. Algoritmin kulku on seuraava:

1. Etsi konveksi monitahokas S_0, jolle $M \subset S_0$. Aseta $k = 0$.
2. Ratkaise lineaarisen optimoinnin ongelma
$\min_{x \in Sk} c^T x.$

Olkoon vastaava optimiratkaisu x_k. Jos $x_k \in M$ niin lopeta. x_k on tällöin ongelman (10) optimiratkaisu. Muuten siirry kohtaan 3.

3. Olkoon j sellainen indeksi, että

$g_j(x_k) = \max g_i(x_k)$.

Aseta

$S_{k+1} = S_k \cap \{x \mid g_j(x_k) + \nabla g_j(x_k)^T(x - x_k) \leq 0\}$.

Siirry kohtaan 2.

2.2.10. Derivaattoja käyttämättömät menetelmät

Edellä mainitut menetelmät ovat kaikki edellyttäneet funktioilta vähintään differentioituvuutta. Joskus tämä on liikaa vaadittu tai derivaattoja ei ole saatavilla. Silloin on käytettävä menetelmiä, joissa derivaattoja ei tarvita (derivative free methods, direct methods). Niistä ehkä tunnetuin on Nelderin-Meadin algoritmi.

Siinä hakupisteitä muodostetaan monitahokkaiden kärkipisteiden avulla. Tällöin hakuun saadaan monipuolisuutta ja vältytään esim. koordinaattiakseleiden suuntien liialliselta korostamiselta. Optimointitehtävänä on

$\min_{x \in \mathbb{R}^n} f(x)$.

Lähtömonitahokas $\subset \mathbb{R}^n$ olkoon kärkien $x_1, \ldots, x_{n+1}$ virittämä, ja x_h se kärki, jossa f saa suurimman arvonsa ja pisteessä x_l pienimmän. Poistetaan x_h ja otetaan jäljelle jääneiden keskiarvo:

$x_c = \frac{1}{n} = \sum_{i=1}^{n+1} x_i,\ \ i \neq h.$

Monitahokasta muunnellaan seuraavilla operaatioilla:

Peilaus. Piste x_h peilataan keskipisteen x_c suhteen kaavalla

$x_p = x_c + \alpha(x_c - x_h)$.

Laajennus. Jos $f(x_p) \leq f(x_l)$, niin

$$x_{ex} = x_c + \gamma(x_p - x_c),$$

missä $\gamma > 1$.

Jos $f(x_{ex}) < f(x_l)$, niin korvataan x_h pisteellä x_{ex}, ja jatketaan näin saadusta muunnetusta monitahokkaasta. Muuten korvataan x_h pisteellä x_p ja jatketaan seuraavasta.

Reduktio. Jos $f(x_p) > f(x_h)$, niin lasketaan kaikki kärjet paitsi x_l uudestaan kaavalla

$$x_i = x_l + \frac{1}{2}(x_i - x_l),\ i \neq l,$$

ja jatketaan seuraavasta:

Kutistus. Jos $f(x_p) > f(x_i)\ \forall i \neq h$, niin kutistetaan kaavalla

$$x_k = x_c + \beta(x_h - x_c),\ 0 < \beta < 1.$$

Vaihdetaan x_h ja x_k ja jatketaan uudella monitahokkaalla.

Kaikissa muissa tapauksissa vaihdetaan x_h ja x_p ja jatketaan uudella monitahokkaalla. Aloitusmonitahokkaaksi valitaan yleensä yksinkertainen säännöllinen monitahokas. Menetelmän huonoja puolia on, että monitahokas voi kutistua dimensioltaan pieneksi eli litistyä. Myöskään konvergenssia ei yleisesti pystytä todistamaan.

2.3. Optimiohjausteoria

Optimiohjausteoria ja variaatiolaskenta ovat varsin keskeisiä matemaattisia menetelmiä taloustieteitten dynaamisissa optimointimalleissa. Molempia menetelmiä käytetään jatkuvasti deterministisissä, jatkuvan ajan dynaamisissa optimointiasetelmissa taloustieteitten eri alueilla. Samankin tyyppisissä malleissa toiset tutkijat käyttävät variaatiolaskentaa ja toiset optimiohjausteoriaa. Käytännön mallitilanteet sisältävät lähes poikkeuksetta jonkinlaisia rajoituksia sekä tilan että ohjauksen suhteen. Tällaisissa tilanteissa optimiohjausteoria saattaa tarjota helpommin muotoiltavan lähestymistavan optimiehtojen käsittelylle kuin variaatiolaskenta. Optimiohjausteoria on myös jossain määrin variaatiolaskentaa yleisempi optimointimenetelmä ja sen puolella tukeudutaan voimakkaasti vektoriesityksen käyttöön. [Salo, 1994]

Optimiohjausteorian kehittäjänä tunnetaan venäläinen Lev Pontryagin [1987]. Hän julkaisi aiheesta ensimmäisen kirjansa jo vuona 1962. Tietokoneiden yleistymisen jälkeen aiheen tutkimus on kehittynyt nopeasti ja nykyisin tietokoneiden avulla voidaan ratkoa jo erittäin monimutkaisiakin ongelmia eri aloilla kuten ilmailu ja avaruus, robotiikka sekä taloustiede.

Optimiohjaus koostuu siihen liittyvien muuttujien differentiaaliyhtälöistä, joiden on tarkoitus minimoida kustannusfunktio. Optimiohjauksen ratkaisu voidaan johtaa Pontryaginin periaatteesta, [Ross, 2009], joka määrittää välttämättömän ehdon optimiohjauksen ongelman ratkaisemiseksi [Pontryagin, 1987].

Dynaamisissa optimointiongelmissa on kyse yli ajan tapahtuvasta (intertemporaalisesta) optimoinnista. Aikadimensio on tällaisissa tapauksissa oleellisesti optimointiin vaikuttava tekijä – nyt tehtävillä päätöksillä on seurausvaikutuksia myöhempinä ajanhetkinä esimerkiksi tulevaisuuden käypien päätösvaihtoehtojen joukon tai tulevaisuuden päätöksien tuottaman hyödyn riippuvuutena tämän hetken päätöksistä.

Yleisesti ottaen optimiohjausongelma voi sisältää useampia tilamuuttujia x_i, i = 1,..., n, vektorina $x = (x_1,..., x_n)$ ja useampia ohjausmuuttujia u_i, i = 1,..., m, vektorina $u = (u_1,..., u_m)$. Varsin kattavan yleisen muodon optimiohjausongelmille antaa tehtävä

$$\max \int_{t_0}^{t_1} f(t, x(t), u(t))dt$$

ehdoin

$$\frac{dx_i}{d_t} = g_i(t,x,u), \qquad i = 1,..., n$$

$$x(t) \in X(t),\ u(t) \in U(x(t),t) \qquad \text{kaikilla } t,\ t_0 \leq t \leq t_1$$

$$x(t_0) = x_0, \qquad x(t_1) \in X_1.$$

Vapaan ajan ongelmissa tarkasteluperiodin loppuhetki t_1 on itsekin päätösmuuttuja ja optimaalisen valinnan kohteena ohjausfunktion u(t) lisäksi. Ajanhetki t_1 voi myös olla ääretön. Ongelman tila x voi kunakin ajankohtana olla rajattu johonkin joukkoon X(t) ja ohjaus u on valittava (mahdollisesti tilasta riippuvasta) ohjausjoukosta U(x, t). Myös lopputila $x(t_1)$ voi olla rajattu johonkin joukkoon.

Edellä esitetyn yleisen, jatkuvan ajan optimiohjausongelman analysointiin yleisimmin käytetyt tekniikat ovat optimiohjausteoria (Pontryaginin maksimiperiaate) ja variaatiolaskenta. Variaatiolaskennan perusongelma on muodoltaan yksinkertaisempi kuin optimiohjausteorian käyttämä asetelma. Diskreettiaikaisissa optimiohjausongelmissa voitaisiin käyttää diskreettiä optimiohjausteoriaa, mutta yleensä kuitenkin käytetään suoraan matemaattisen optimoinnin antamia Karushin-Kuhnin-Tuckerin ehtoja. Erityisesti numeerista ratkaisua tavoiteltaessa, mutta myös sopivan tyyppisissä diskreettiaikaisissa tehtävissä, dynaaminen ohjelmointi on kätevä ratkaisutekniikka. Dynaamisesta ohjelmoinnista on myös jatkuva-aikainen versio, mutta se ei yleensä ole yhtä kätevä käyttää kuin optimiohjausteoria tai variaatiolaskenta. Sen sijaan stokastisissa ongelmissa dynaaminen ohjelmointi on osoittautunut käyttökelpoisimmaksi tekniikaksi. [Salo, 1994]

Optimiohjausteorian perusongelma (Pontryaginin maksimiperiaate)

Optimiohjausteorian perusongelmaa kutsutaan myös Pontryaginin maksimiperiaatteeksi [Kamien and Schwartz, 2012]. Paloittain jatkuvan ohjausmuuttujan $u(t) = (u_1(t),..., u_m(t))$ (vektori) ja jatkuvan sekä paloittain differentioituvan tilamuuttujan $x(t) = (x_1(t),...,x_n(t))$ (vektori), jotka on määritelty koko välillä $[t_0\ t_1]$, avulla

$$\max \int_{t_0}^{t_1} f(t, x(t), u(t))dt$$

ehdoin

$$x'_i(t) = g_i(t, x(t), u(t)), \qquad i = 1,..., n,$$

$$x_i(t_0) = x_{i0}, \qquad i = 1,..., n, \qquad (x_{i0} \text{ kiinnitetty}),$$

loppuehdoilla

$x_i(t_i) = x_{i1}$, $i = 1,..., p$,

$x_i(t_i) \geq x_{it}$, $i = p + 1,..., q$, (x_{i1}, $i = 1,...,q$, kiinnitettyjä)

$x_i(t_i)$ vapaa, $i = q + 1,..., n$,

ja tilamuuttujan rajoituksella

$u(t) \in U$, $U \subset \mathbb{R}^m$ on kiinteä joukko.

Funktioiden f ja g_i sekä niiden ensimmäisten osittaisderivaattojen $\partial f/\partial x$ ja $\partial g_i/\partial x_j$ oletetaan olevan jatkuvia (t, x, u):n funktioita kaikilla i = 1,..., n ja j = 1,..., n.

Jotta x*(t) olisi optimirata ja u*(t) optimiohjaus yllä kuvatulle ongelmalle, on oltava sellaiset vakio λ_0 ja jatkuva vektorifunktio $\lambda(t) = (\lambda_1(t),..., \lambda_n(t))$, jossa kaikilla $t_0 \leq t \leq t_1$ on voimassa $(\lambda_0, \lambda(t)) \neq (0, 0)$, että kaikilla $t_0 \leq t \leq t_1$ pätee

$H(t, x^*(t), u, \lambda(t)) \leq H(t, x^*(t), u^*(t), \lambda(t))$,

missä Hamiltonin funktio H on määritelty λ avulla seuraavasti:

$H(t, x, u, \lambda) = \lambda_0 f(t, x, u) + \sum_{i=1}^{n} \lambda_i g_i(t, x, u)$.

Poikkeuksena ovat u*(t):n epäjatkuvuuskohdat

$\lambda'_i(t) = -\partial H(t, x^*(t), u^*(t), \lambda(t))/\partial x_i$, $i = 1,..., n$.

Edelleen on voimassa

$\lambda_0 = 1$ tai $\lambda_0 = 0$

sekä seuraavat transversaalisuusehdot:

$\lambda_i(t_1)$ ilman ehtoja, $i = 1,..., p$,

$\lambda_i(t_1) \geq 0$ (= 0, jos $x_i^*(t1) > x_{i1}$), $i = p + 1,..., q$,

$\lambda_i(t_1) = 0$, $i = q + 1,..., n$.

Optimiohjausteorian ongelmia käsiteltäessä pitää kiinnittää huomiota mm. tila- ja ohjausmuuttujien välttämättömiin ja riittäviin ehtoihin, yhtälö- ja epäyhtälörajoitteisiin sekä Hamiltonin funktion muodostamiseen sekä erilaisiin transversaalisuusehtoihin. Lisäksi äärettömän horisontin tapaus tuo omat erikoisongelmansa optimaalisuuden ratkaisemiseksi. [Salo, 1994]

2.4. Dynaaminen optimointi

Dynaamisen optimoinnin englanninkielinen nimi ”Dynamic programming” viittaa vahvasti ohjelmointiin, mutta tietojenkäsittelytieteen lisäksi sillä on vahva asema mm. matematiikassa, taloustieteessä ja bioinformatiikassa. Dynaaminen optimointi on luonteva jatke variaatiolaskennalle ja optimiohjausteorialle. Se tutkii tehtäviä, joissa optimointistrategia perustuu ongelman jakamiseen pienemmiksi osaongelmiksi ja lopullinen ratkaisu saadaan yhdistämällä osaongelmien vastaukset.

Dynaamisen optimoinnin perusteita kehitti 1950-luvulla Richard Bellman ja hänen mukaansa on nimetty yksi teorian tärkeimmistä osista eli Bellmanin yhtälö [Bellman, 1957], joka määrittää optimaalisuudelle välttämättömyysehdon. Bellmanin yhtälö liitetään yleisesti diskreetin ajan optimointiongelmiin, kun taas Hamiltonin-Jacobin-Bellmanin (HJB) yhtälöä käytetään jatkuvan ajan optimointiongelmissa. HJB-yhtälö on lisäys William Rowan Hamiltonin ja Carl Gustav Jacob Jacobin kehittämälle Hamiltonin-Jacobin yhtälölle. Melkein kaikki optimiohjausteorialla ratkaistavat ongelmat voidaan ratkaista myös analysoimalla sopivaa Bellmanin yhtälöä.

Bellmanin kehittämän optimaalisuuden periaatteen mukaan seuraavaan tilaan johtavien päätösten tulee aina olla optimaalisia riippumatta aiemmista päätöksistä. Periaatetta hyödyntämällä voidaan rajoittaa käsiteltävien potentiaalisten optimiohjausstrategioiden lukumäärää.

Yleisesti ottaen optimointitehtävän tavoitteena on jonkin funktion minimointi tai maksimointi, esimerkiksi matkustusajan tai kulujen minimointi tai tuoton maksimointi. Kyseistä tavoitetta kuvaava matemaattinen funktio on siis kohdefunktio (objective function).

Dynaaminen optimointi pilkkoo monimutkaisen etenevässä ajassa tapahtuvan ongelman pienemmiksi osaongelmiksi, jotka ratkaistaan eri aikoina. Jokaiseen hetkeen liittyvästä informaatiosta, jonka avulla tehdään päätöksiä, käytetään nimitystä tila [Bellman, 1957]. Näitä tilamuuttujia voi olla useampi kuin yksi.

Rajoitusmuuttuja vaikuttaa tilamuuttujan muutokseen siirryttäessä seuraavaan ajanhetkeen. Myös näitä muuttujia voi olla useampi kuin yksi. Dynaamisella optimoinnilla pyritään kuvaamaan optimaalinen suunnitelma, jolla löydetään rajoitteille sääntö millä tahansa tilan arvolla. Tätä sääntöä kutsutaan ohjausfunktioksi (policy function) [Bellman, 1957].

Lopulta saadaan optimaalinen sääntö tehtävälle päätökselle, jonka avulla kohdefunktio saa parhaan mahdollisen arvon. Kun tämä arvo esitetään tilan funktiona, niin puhutaan arvofunktiosta (value function).

Bellmanin työn tuloksena dynaamisen optimoinnin ongelma diskreetissä ajassa voidaan ratkaista rekursiivisesti askel askeleelta, kun ratkaistaan peräkkäisten ajanjaksojen arvofunktioiden välinen suhde. Tämän suhteen kuvaamiseksi hän siis kehitti (myöhemmin Bellmanin yhtälöksi kutsutun) yhtälön:

$$V(x_0) = \max_{a0}\{F(x_0, a_0) + \beta V(x_1)\}$$

ehdoin

$$a_0 \in \Gamma(x_0),\ x_1 = T(x_0, a_0).$$

Yhtälöstä saadaan selkeämpi muoto merkinnöillä

$$V(x) = \max_{a \in \Gamma(x)} \{F(x, a) + \beta V(T(x, a))\}.$$

Kaavan merkinnöissä

x_t on tila ajassa t,

x_0 on alkutila ajassa t = 0,

$a_t \in \Gamma(x_t)$ tilasta riippuva mahdollisten toimintojen joukko,

a_t on toiminto, joka edustaa yhtä tai useampaa rajoitusmuuttujaa,

$T(x, a)$ on uusi tila toiminnon a:n jälkeen,

$F(x, a)$ on payoff, kun on suoritettu toiminto a tilassa x,

$0 < \beta < 1$ on korjaustermi,

$V(x_0)$ on optimaalinen arvo eli arvofunktio, joka voidaan saavuttaa maksimoimalla kohdefunktio rajoitukset huomioonottaen.

Bellmanin yhtälöä sanotaan funktionaaliksi, koska sen ratkaiseminen merkitsee tuntemattoman funktion V löytämistä. Arvofunktion laskemisella löydetään myös funktio $a(x)$, joka siis kuvaa optimaalista toimintaa tilan funktiona eli $a(x)$ on ohjausfunktio.

Dynaamisen optimoinnin luonteeseen kuuluu ratkaista kukin osaongelma vain kerran, jolloin laskutoimitusten kokonaislukumäärä pienenee. Tämä on erittäin hyödyllinen ominaisuus, kun toistettavien osaongelmien määrä kasvaa eksponentiaalisesti syötteen koon funktiona.

Dynaamista optimointia käytetään sellaisissa algoritmeissa, jotka sisältävät optimaalisen alirakenteen (optimal substructure) ja päällekkäisiä osaongelmia (overlapping subproblems). Tällaiset ”ahneet” algoritmit ovat nopeampia kuin yksinkertaisemmat algoritmit, mutta ne eivät välttämättä johda optimaaliseen ratkaisuun.

Dynaamisen optimoinnin suurin ongelma on algoritmin suuri tilan tarve. Bellman itse kutsui ongelmaa dimensionaalisuuden kiroukseksi [Betts, 1998].

2.5. Lentoradan optimointi

Lentoratojen optimointia lähdettiin kehittämään tietokoneiden yleistyessä 1950-luvulla variaatiolaskennan ja optimiohjauksen teorioiden pohjalta. Optimoinnin tavoitteena on löytää sellainen lentorata, joka on optimaalinen suhteessa tavoitteisiin, eikä riko sille asetettuja rajoituksia. Periaatteessa kaikki numeeriset lentoradan optimointimenetelmät sisältävät jonkin tyyppistä iteraatiota äärellisellä joukolla tuntemattomia muuttujia ja Newtonin menetelmä on niissä merkittävässä asemassa [Betts, 1998].

Lentoradan optimoinnin ongelma on periaatteessa sama kuin optimiohjauksen (Optimal control theory) ongelman ratkaiseminen [Ross, 2009] ja siinä sovelletaankin samoja algoritmeja kuin muissakin optimointiongelmissa. 1950-luvulla tutkimus keskittyi rakettien työntövoiman optimoimiseen

ja suihkukoneiden nousukyvyn maksimoimiseen. Alkuaikoina vaikeuksia aiheutti yksittäisten alikaarien (singular subarcs) tapaus. Näissä ongelmissa Hamiltonin rajoitusmuuttujan termi menee nollaan äärelliseksi ajaksi, jolloin optimiohjauksen suora ratkaiseminen tulee mahdottomaksi. Tässä tapauksessa Hamiltonin funktio on muotoa $H(u) = \emptyset(x, \lambda, t)u + ...$ ja rajoitusfunktiolla on ylä- ja alarajat, kun $a \leq u(t) \leq b$. Jotta $H(u)$ saataisiin minimoitua, pitää u:n olla mahdollisimman suuri tai pieni $\emptyset(x, \lambda, t)$:n etumerkistä riippuen, eli

$$u(t) = \begin{cases} b, \emptyset(x,\lambda,t) < 0 \\ ?, \emptyset(x,\lambda,t) = 0 \\ a, \emptyset(x,\lambda,t) > 0. \end{cases}$$

Niin kauan kun $\emptyset$ on positiivinen tai negatiivinen ja 0 vain hetkittäin, on ratkaisu suoraviivainen ”bang-bang”-ohjaus, joka vaihtelee a:sta b:hen, kun $\emptyset$ vaihtelee positiivisesta negatiiviseksi. Tehtävä on ongelmallinen siis silloin, kun $\emptyset$ pysyy arvossa 0. Tässä singulaariohjauksen tapauksessa optimaalinen lentorata seuraa yksittäistä alikaarta (singular subarc) ja vaihtoehtoja ratkaisulle on useampia. Optimaalisen ratkaisun löytämiseksi piti käydä vaihtoehdot läpi manuaalisesti kunnes ongelmalle löydettiin Kelleyn ehto:

$$(-1)^k \frac{\partial}{\partial u} [(\frac{d}{dt})^{2k} H_u] \geq 0, \text{ k} = 0, 1, ...,$$

joka takaa yksittäisen alikaaren optimaalisuuden. Tätä ehtoa sanotaan myös yleistetyksi Legendrenin-Clebschin ehdoksi [Bryson, 1975].

Lentoradan optimointimenetelmät voidaan karkeasti ottaen jakaa epäsuoriin ja suoriin metodeihin [Betts, 1998]. Lisäksi on tietenkin myös kehitetty tekniikoita, joissa nämä kaksi ratkaisutapaa yhdistyvät. Ensimmäiseen ryhmään kuuluvat optimiohjaukseen liittyvät metodit, joissa ratkaisut saadaan analyyttisillä tai numeerisilla menetelmillä. Jälkimmäisessä ryhmässä on puolestaan sellaisia metodeja, jotka antavat hyvän arvion optimiohjauksen ongelmaan epälineaarisen optimoinnin avulla. Epäsuora metodi tuottaa vastauksen, jossa ”total differential of the performance measure” on nolla. Suoralla metodilla vastauksena saadaan arvo, joka on joko pienempi tai suurempi kuin mikään muu naapuruston arvo eli paikallinen minimi tai maksimi. Näiden kahden metodin käyttö on yhdistetty ”covector mapping” periaatteessa [Ross and Karpenko, 2012].

Optimiohjauksen ongelma voi tunnetusti olla dimensionaalisuudeltaan ääretön, kun taas epälineaarisen optimoinnin tehtävät ovat dimensionaalisuudeltaan äärellisiä. Optimiohjauksen numeeriset metodit voivat tuottaa parhaat vastaukset, mutta suppeneminen voi myös olla ongelmallista. Jos alustava arvaus on onnistunut, on suppeneminen nopeaa, mutta huonolla arvauksella etsintä epäonnistuu. Avaruusalusten laukaisussa optimiohjaus toimii hyvin siihen liittyvän erittäin tiukan toleranssin ansiosta, mutta löyhemmän kontrollin ympäristöissä se ei ole yhtä hyödyllinen.

Analyyttinen ratkaisu optimiohjaukselle sisältää usein mittavia approksimaatioita, mutta tuottaa kuitenkin hyödyllisiä algoritmeja. Ohlmeyerin ja Phillipsin [2006] ratkaisussa tehdään lineaarisia oletuksia ja päästään lähelle optimaalista lentorataa. Toinen analyyttinen ratkaisumalli, Iterative Guidance Mode (IGM) -algoritmi, on käytössä Saturn V -raketin laukaisussa. Se on variaatiolaskennan ratkaisu ”two-point boundary value” -ongelmaan, kun raketti laukaistaan kiertoradalle. Ratkaisu

edellyttää painovoiman kiihtyvyyden approksimoimista vakiollisena vektorina sekä ratkaisun iterointia riittävän tarkkuuden saavuttamiseksi.

Parametrien optimoimiseksi on useita numeerisia menetelmiä. Näistä yksinkertaisimmat käyttävät ”gradient descent” -tekniikkaa. Suppenemista tehostavia toisen asteen metodeja on myös käytössä kuten Newtonin–Raphsonin menetelmä (Newtonin menetelmä). Kvasi-Newtonin menetelmää käyttämällä vältytään edellisessä vaadittavalta Hessen matriisin laskemiselta.

Broydenin–Fletcherin–Goldfarbin–Shannonin (BFGS) algoritmi ja ”sequential quadratic programming” (SQP) ovat iteratiivisia menetelmiä epälineaariselle optimointitehtävälle. Simplex-algoritmia käytettiin aluksi avaruusalusten paluulentoradan määrittämiseen, mutta myöhemmin sitä on tehokkuutensa ja luotettavuutensa vuoksi käytetty laajasti muissakin rakettien lentoratojen optimointiongelmissa [Williams and Tucker, 1973].

Monitasoinen optimointitekniikka on käyttökelpoinen vaihtoehto silloin, kun ollaan tekemisissä kompleksisten payoff-funktioiden kanssa. Esimerkiksi Phillips ja Drake [2000] ovat tutkineet tätä tekniikkaa ohjuskäyttöön liittyen.

2.5.1. Yleinen lentoradan optimoinnin ongelma

Yleisellä tasolla lentoradan optimointiongelma voidaan kuvata seuraavasti [Betts, 1998]:
Ongelma määritellään kokoelmana vaiheita, joita on N kappaletta. Riippumaton muuttuja t vaiheelle k on määritelty $t_0^{(k)} \le t \le t_f^{(k)}$. Monissa tapauksissa t kuvaa aikaa ja vaiheet ovat peräkkäisiä eli $t_0^{(k+1)} = t_f^{(k)}$, mutta kumpikaan ominaisuus ei ole välttämätön. Vaiheen k sisällä järjestelmän dynamiikkaa kuvataan joukolla dynaamisia muuttujia

$$\mathrm{z} = \begin{bmatrix} \mathrm{y}^{(k)} & (t) \\ \mathrm{y}^{(k)} & (t) \end{bmatrix},$$

jotka koostuvat $n_y^{(k)}$ tilamuuttujista ja $n_u^{(k)}$ rajoitemuuttujista. Lisäksi dynamiikkaan voidaan liittää $n_p^{(k)}$ parametrit $\mathbf{p}^{(k)}$, jotka eivät ole riippuvaisia muuttujasta t. Tyypillisesti järjestelmän dynamiikka määritetään joukosta tavallisia differentiaaliyhtälöitä eksplisiittiseen muotoon kirjoitettuna, johon viitataan tila- tai järjestelmäyhtälöinä

$$\dot{\mathbf{y}} = \mathbf{f}[\mathbf{y}(t), \mathbf{u}(t), \mathbf{p}(t)],$$

missä $\mathbf{y}$ on n_y ulotteinen tilavektori. Aloitusehdot hetkellä t_0 on määritelty muodossa

$$\psi_{0l} \le \psi[\mathbf{y}(t_0), \mathbf{u}(t_0), \mathbf{p}, t_0] \le \psi_{0u},$$

missä $\psi[\mathbf{y}(t_0), \mathbf{u}(t_0), \mathbf{p}, t_0] \equiv \psi_0$ ja lopetusehdot hetkellä t_f on määritelty muodossa

$$\psi_{fl} \le \psi[\mathbf{y}(t_f), \mathbf{u}(t_f), \mathbf{p}, t_f] \le \psi_{fu},$$

missä $\psi[\mathbf{y}(t_f), \mathbf{u}(t_f), \mathbf{p}, t_f] \equiv \psi_f$. Lisäksi ratkaisun pitää täyttää algebrallisen polun ehdot, jotka ovat muotoa

$$\mathbf{g}_l \le \mathbf{g}[\mathbf{y}(t), \mathbf{u}(t), \mathbf{p}(t)] \le \mathbf{g}_u,$$

missä $\mathbf{g}$ on n_g ulotteinen vektori ja myös ala- ja ylärajat tilamuuttujille asetetut ehdot

$$\mathbf{y}_l \le \mathbf{y}(t) \le \mathbf{y}_u,$$

sekä rajoitemuuttujille asetetut ehdot

$\mathbf{u}_l \leq \mathbf{u}(t) \leq \mathbf{u}_u$.

Tässä yhteydessä saattaa olla hyödyllistä arvioida myös lausekkeita

$\int_{t0}^{tf} \mathbf{q}[\mathbf{y}(t), \mathbf{u}(t), \mathbf{p}, t]dt$.

Yleisesti ottaen vaiheen sisällä tutkittaviin funktioihin, eli

$$\mathbf{F}(t) = \begin{bmatrix} \mathbf{f}[\mathbf{y}(t), \mathbf{u}(t), \mathbf{p}, t] \\ \mathbf{g}[\mathbf{y}(t), \mathbf{u}(t), \mathbf{p}, t] \\ \mathbf{q}[\mathbf{y}(t), \mathbf{u}(t), \mathbf{p}, t] \end{bmatrix}$$

viitataan jatkuvien funktioiden vektorina. Samalla tapaa tietyssä pisteessä tutkittaviin funktioihin, kuten reunaehtoihin $\psi[\mathbf{y}(t_0), \mathbf{u}(t_0), t_0]$ ja $\psi[\mathbf{y}(t_f), \mathbf{u}(t_f), t_f]$, viitataan pistefunktioina.

Yleinen optimiohjauksen ongelma määrittää $n_u^{(k)}$-dimensionaaliset $\mathbf{u}^{(k)}(t)$ rajoitevektorit ja parametrit $\mathbf{p}^{(k)}$ minimoidakseen kohdefunktion

$J = \emptyset[\mathbf{y}(t_0^{(1)}), t_0^{(1)}, \mathbf{y}(t_f^{(1)}), \mathbf{p}^{(1)}, t_f^{(1)}, \ldots, \mathbf{y}(t_0^{(N)}), t_0^{(N)}, \mathbf{y}(t_f^{(N)}), \mathbf{p}^{(N)}, t_f^{(N)}]$.

Tässä määrittelyssä vaiheella (phase) tarkoitetaan lyhyempää ajanjaksoa ja siitä käytetään myös nimitystä kaari (arc). Differentiaaliyhtälöt eivät muutu vaiheen aikana, mutta voivat muuttua vaiheesta toiseen. Vaiheet mahdollistavat muutokset dynamiikassa lentoradan aikana. Vaiheen rajaa kutsutaan tapahtuma- tai risteyspisteeksi (event/junction point). Rajaehtoa, joka siis määrittää vaiheen päättymisen, sanotaan tapahtumakriteeriksi (event criterion) ja hyvin määritellyllä ongelmalla voi olla vain yksi kriteeri jokaisessa tapahtumassa. Monimutkaisten lentoratojen simulaatioissa voidaan vaiheet normaalisti linkittää yhteen pakottamalla tilat jatkuviksi eli $\mathbf{y}(t_f^{(1)}) = \mathbf{y}(t_0^{(2)})$. Differentiaaliyhtälöt, $\dot{\mathbf{y}} = \mathbf{f}[\mathbf{y}(t), \mathbf{u}(t), \mathbf{p}, t]$, on kirjoitettu ilmailualan standardien mukaisesti ensimmäisen asteen yhtälöiden eksplisiittisenä järjestelmänä eli derivaatta esiintyy ainoastaan yhtälön vasemmalla puolella. Kohdefunktio J on kirjoitettu vaiheiden lopussa arvioituina suureina (terms of quantities) eli ns. Mayerin muodossa [Vapnyarskii, 2011]. Jos kohdefunktio sisältää ainoastaan integraalin

$\int_{t0}^{tf} \mathbf{q}[\mathbf{y}(t), \mathbf{u}(t), \mathbf{p}, t]dt$,

niin siihen viitataan Lagrangen ongelmana [Vapnyarskii and Tikhomirov, 2011]. Kun molemmat termit esiintyvät, niin käytetään nimitystä Bolzan ongelma [Vapnyarskii, 2011]. Lauseke voidaan saattaa Mayerin muotoon sekä Lagrangen että Bolzan ongelmasta.

Betts [1998] luokittelee numeerisia menetelmiä sen mukaan, minkälaisella proseduurilla ne algoritmissaan toteuttavat funktion arvioimisen ja erityisesti funktion generoimisen (function generator). Hän esittää viisi menetelmää: direct shooting, indirect shooting, multiple shooting, indirect transcription ja direct transcription. Kaikissa menetelmissä annetaan muuttujat syötteenä funktiogeneraattorille ja tuloksena saadaan kohdefunktio sekä rajoitteet.

2.5.2. Suora tähtäysmenetelmä

Suora tähtäysmenetelmä eli ”direct shooting” on yksi käytetyimmistä algoritmeista raketin laukaisuissa ja kiertoradan toiminnoissa, koska niissä on suhteellisen vähän optimoinnin muuttujia (algoritmi 1). POST (Program to Optimize Simulated Trajectories) on suosittu tätä menetelmää käyttävä ohjelmisto ja se tai jokin vastaava on käytössä useimmissa suurissa ilmailualan yrityksissä. Tässä

menetelmässä muuttujiksi valikoituu alkuehtojen, loppuehtojen ja parametrien alijoukko eli jokaiselle vaiheelle määritetään

$$X^{(k)} = \{\mathbf{y}(t_0), \mathbf{p}, t_0, \mathbf{y}(t_f), t_f\}.$$

Tästä muodostuu muuttujien täydellinen joukko

$$\mathrm{x} \subset \{X^{(1)}, X^{(2)}, \ldots, X^{(N)}\}.$$

Optimointitehtävän rajoitteet ja kohdefunktio ovat siis määreitä, jotka arvioidaan vaiheiden rajakohdissa, jolloin

$$\mathbf{c}(\mathbf{x}) = \begin{bmatrix} \psi^{(1)}[\mathbf{y}(\mathrm{t}_0), \mathbf{p}, \mathrm{t}_0] \\ \psi^{(1)}[\mathbf{y}(\mathrm{t}_f), \mathbf{p}, \mathrm{t}_f] \\ \vdots \\ \psi^{(N)}[\mathbf{y}(\mathrm{t}_0), \mathbf{p}, \mathrm{t}_0] \\ \psi^{(N)}[\mathbf{y}(\mathrm{t}_f), \mathbf{p}, \mathrm{t}_f] \end{bmatrix}.$$

Alkuarvo-ongelmassa, Initial Value Problem (IVP), on tehtävänä laskea $\mathbf{y}(t_f)$ jollekin arvolle $t_0 < t_f$ joka toteuttaa ehdon

$$\dot{\mathbf{y}} = \mathbf{f}[\mathbf{y}(t), t]. \qquad (11)$$

Syöte: **x**
do for (jokaiselle vaiheelle) $k = 1, N$
  Alusta Vaihe k:
    $\mathbf{y}^{(k)}(t_0)$, $\mathbf{p}^{(k)}$, $t_0^{(k)}$
  Rajoitteen Arviointi:
    Laske $\psi^{(k)}[\mathbf{y}(t_0), \mathbf{p}, t_0]$
  Alkuarvo-ongelma:
    Annettu t_f laske $\mathbf{y}^{(k)}(t_f)$, eli ratkaise (11)
  Rajoitteen Arviointi:
    Laske $\psi^{(k)}[\mathbf{y}(t_f), \mathbf{p}, t_f]$
end do
Lopeta Lentorata
  Laske tavoite $F(\mathbf{x})$, $\mathbf{c}(\mathbf{x})$
Tulosta: $F(\mathbf{x})$, $\mathbf{c}(\mathbf{x})$

Algoritmi 1. Suora tähtäysmenetelmä

2.5.3. Epäsuora tähtäysmenetelmä

Epäsuoran tähtäysmenetelmän (indirect shooting) yksinkertaisimmassa tapauksessa muuttujiksi valikoituu alijoukko optimiohjauksen välttämättömien ehtojen raja-arvoja (algoritmi 2). Tässä tapauksessa muuttujat ovat muotoa $\mathrm{x} = \{\boldsymbol{\lambda}(t_0), t_f\}$.

Nyt rajoitteiksi saadaan ehto

$$\mathbf{c}(\mathbf{x}) = \begin{bmatrix} \psi[\mathbf{y}(t), \mathbf{p}, t] \\ \boldsymbol{\lambda}(t) - \Phi_t^T \\ (\Phi_t + H) \end{bmatrix}, t=t_f. \quad (12)$$

Suurin ero suoran ja epäsuoran tähtäysmenetelmä välillä on ohjausfunktion määrittelyssä. Epäsuorassa menetelmässä ohjaus määritetään maksimiperiaatteella jokaisessa ajan pisteessä. Näin ollen $\boldsymbol{\lambda}(t_0)$:n arvoista tulee $\mathbf{p}$:n sijaan optimiohjausfunktion määrittäviä parametreja. Epäsuoran menetelmän ongelma on sen herkkyys aloituskohdan valitsemiselle ja kirjallisuudessa onkin jouduttu keskittymään myös hyvän aloitusmetodin löytämiseen. Suurin ongelma on kuitenkin välttämättömien ehtojen johtaminen, koska realistisissa lentoratojen simuloinneissa epäyhtälöt, polun rajoitteet ja rajaehdot voivat kaikki olla monimutkaisia matemaattisia lausekkeita.

Syöte: $\mathbf{x}$
do for (jokaiselle vaiheelle) $k = 1, N$
  Alusta Vaihe k:
    $\mathbf{y}^{(k)}(t_0)$, $\boldsymbol{\lambda}^{(k)}(t_0)$, $\mathbf{p}^{(k)}$, $t_0^{(k)}$
  Alkuarvo-ongelma:
    Annettu t_f laske $\mathbf{y}^{(k)}(t_f)$, $\boldsymbol{\lambda}^{(k)}(t_f)$
  Rajoitteen Arviointi:
    Arvioi (12)
    $$\mathbf{c}(\mathbf{x}) = \begin{bmatrix} \psi[\mathbf{y}(t), \mathbf{p}, t] \\ \boldsymbol{\lambda}(t) - \Phi_t^T \\ (\Phi_t + H) \end{bmatrix}, t=t_f.$$
end do
Lopeta Lentorata
Tulosta: $\mathbf{c}(\mathbf{x})$

Algoritmi 2. Epäsuora tähtäysmenetelmä

2.5.4. Monimaalimenetelmä

Sekä suora että epäsuora tähtäysmenetelmä kärsivät yleisestä monimutkaisuudesta. Molempien ongelmana on myös alkuvaiheessa mahdollisesti tapahtuvien pienten virheiden suuri vaikutus lopputulokseen. Monimaalimenetelmä (Multiple Shooting, monipisteammunta) määritellään yksinkertaisimmassa muodossaan seuraavasti: lasketaan tuntemattomat lähtöarvot $\mathbf{v}(t_0) = \mathbf{v}_0$ siten, että reunaehto $0 = \phi[\mathbf{v}(t_f), t_f]$ pätee jollekin arvolle $t_0 < t_f$, joka toteuttaa

$\dot{\mathbf{v}} = \mathbf{f}[\mathbf{v}(t), t]$.

Monimaalimenetelmän ideana on jakaa lentorata lyhyisiin osiin. Niinpä kokonaisaika jaetaan pienempiin jaksoihin

$t_0 < t_1 < \ldots < t_M = t_f$.

Yhdistämällä kaikkien jaksojen muuttujat saadaan muuttujajoukko

$\mathbf{x} = \{\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_{M-1}\}$.

Jotta varmistetaan jaksojen yhdistyminen ehtojen rajoissa, asetetaan rajoitteille

$$\mathbf{c}(\mathbf{x}) = \begin{bmatrix} \mathbf{v_1} - \bar{\mathbf{v}}_\mathbf{0} \\ \mathbf{v_2} - \bar{\mathbf{v}}_\mathbf{1} \\ \vdots \\ \Phi[\mathbf{v_M}, \boldsymbol{t_f}] \end{bmatrix} = 0. \qquad (13)$$

Menetelmä lisää muuttujien ja rajoitteiden määrää jokaisessa jaksossa ja näin ollen kasvattaa Newtonin iteraation ongelman kokoa. Optimointitehtävän koko on $n = n_v M$, missä n_v ilmoittaa dynaamisten muuttujien **v** lukumäärän ja M jaksojen lukumäärän. Tilannetta helpottaa kuitenkin Newtonin menetelmän etsintäsuunnan laskemisessa käytettävän Jacobin matriisin **A** harvuus, sillä vain Mn_v^2 elementtiä on nollasta eriäviä. Yleisesti ottaen alkuperäisessä ongelmassa esiintyneet vaiheet pilkotaan tässä menetelmässä pienemmiksi jaksoiksi (algoritmi 3).

Monimaalimenetelmä voidaan sisällyttää sekä suoraan että epäsuoraan tähtäysmenetelmään. Näissä ero syntyy dynaamisten muuttujien **v**, dynaamisen systeemin $\dot{\mathbf{v}} = \mathbf{f}[\mathbf{v}(t), t]$ ja reunaehtojen $0 = \phi[\mathbf{v}(t_f), t_f]$ määrittämisessä. Suorassa monimaalimenetelmässä dynaamiset muuttujat **v** määritetään tilalla ja ohjauksella (**y**, **u**). Epäsuorassa monimaalimenetelmässä puolestaan dynaamiset muuttujat **v** sisältävät tilan, ohjauksen ja liittotilamuuttujat (**y**, **u**, **λ**). Suorassa metodissa optimointitehtävän muuttujien **x** ja rajoitteiden **c**(**x**) lukumäärä on sama, mutta epäsuorassa metodissa ei näin tarvitse olla.

Multiple Shooting -menetelmän suurin etu on sen robustisuus, ja näin ollen se parantaa sekä suoran että epäsuoran tähtäysmenetelmän toimintavarmuutta. Mielenkiintoinen ominaisuus on myös soveltuvuus rinnakkaisprosessoinnin hyödyntämiseen. Jokainen jakso ja/tai vaihe voidaan nimittäin toteuttaa yhtäaikaisesti rinnakkaisten prosessorien avulla. Menetelmää kutsutaan joskus myös nimellä parallel shooting.

Syöte: **x**
do for (jokaiselle vaiheelle) $k = 1, N$
  Alusta Vaihe k:
  do for (jokaiselle segmentille) $j = 0, M - 1$
    Alusta Segmentti $j + 1$:
      $\mathbf{v}_j, t_j$
    Alkuarvo-ongelma:
      Annettu t_{j+1} laske $\bar{\boldsymbol{v}}_j$
    Rajoitteen Arviointi:
      tallenna $\mathbf{v}_{j+1} - \bar{\boldsymbol{v}}_j$, kaava (13)
  end do
    tallenna $\phi[\mathbf{v}_M, t_f]$, kaava (13)
end do
Lopeta Lentorata
Tulosta: $\mathbf{c}(\mathbf{x})$ (ja $F(\mathbf{x})$)

Algoritmi 3. Monimaalimenetelmä

2.5.5. Epäsuora kollokaatiomenetelmä

Epäsuora kollokaatiomenetelmä (indirect transcription, collocation) on kehitetty ratkaisemaan kahden pisteen reuna-arvotehtäviä, joihin törmätään mm. epäsuorien metodien välttämättömiä ehtoja ratkaistaessa. Nämä kollokaatiomenetelmät voivat olla erittäin tehokkaita, kun ratkaistaan useiden pisteiden reuna-arvotehtäviä kuten lentoratojen optimointi (algoritmi 4).

Menetelmän lähtökohta on sama kuin monimaalimenetelmässäkin, eli lasketaan tuntemattomat lähtöarvot $\mathbf{v}(t_0) = \mathbf{v}_0$ siten, että reunaehto $0 = \phi[\mathbf{v}(t_f), t_f]$ pätee jollekin arvolle $t_0 < t_f$, joka toteuttaa ehdon

$$\dot{\mathbf{v}} = \mathbf{f}[\mathbf{v}(t), t].$$

Kuten monimaalimetodissa myös tässä lentoradan kokonaisaika jaetaan pienempiin ajanjaksoihin

$$t_0 < t_1 < \ldots < t_M = t_f.$$

Rajoitteet ovat muotoa

$$0 = \mathbf{v}_{j+1} - \bar{\mathbf{v}}_j$$
$$= \mathbf{v}_{j+1} - (\mathbf{v}_j + h_j\mathbf{f}_j), \text{ (Defect-rajoite)}$$

missä $t_{j+1} = h_j + t_j$ kaikille jaksoille $j = 0,\ldots, (M - 1)$.

Kollokaatiomenetelmän Defect-rajoitteen tyydyttämiseksi käytetään usein Hermiten-Simpsonin metodia:

$$0 = \mathbf{v}_{j+1} - \mathbf{v}_j - (h_{j+1}/6)[\mathbf{f}_{j+1} + 4\bar{\mathbf{f}}_{j+1} + \mathbf{f}_j] = \varsigma_j. \qquad (14)$$

Kollokaatiomenetelmän ratkaisu on paloittain jatkuva polynomi, joka toteuttaa tavalliset differentiaaliyhtälöt niin sanotuissa kollokaatiopisteissä $t_j \leq t \leq t_{j+1}$.

Syöte: **x**
do for (jokaiselle vaiheelle) $k = 1, N$
  Alusta Vaihe k:
  do for (jokaiselle verkon pisteelle) $j = 0, M - 1$
    Rajoitteen Arviointi: tallenna
      discretization defect, esim. (14)
  end do
    tallenna $\phi[\mathbf{v}_M, t_f]$, kaava (13)
end do
Lopeta Lentorata
Tulosta: **c**(**x**)

Algoritmi 4. Epäsuora kollokaatiomenetelmä

2.5.6. Suora kollokaatiomenetelmä

Direct Transcription -menetelmää (algoritmi 5), kuten muitakin suoria menetelmiä, voidaan soveltaa ilman välttämättömien ehtojen, kuten transversaalisuus ja maksimiperiaate, eksplisiittistä johtamista. Toisekseen, poikkeuksena muista edellä esitellyistä metodeista, se ei vaadi kaarien järjestyksen (arc sequence) selvittämistä etukäteen.

Myös tässä algoritmissa kokonaisaika jaetaan ensin pienempiin osiin

$t_0 < t_1 < \ldots < t_M = t_f$,

jolloin muuttujat saavat arvonsa tilasta ja ohjauksesta ruudukon pisteissä, eli

$\mathbf{x} = \{\mathbf{y}_0, \mathbf{u}_0, \mathbf{y}_1, \mathbf{u}_1, \ldots, \mathbf{y}_M, \mathbf{u}_M\}$.

Lopulta saadaan siis

$$\mathbf{c}(\mathbf{x}) = \begin{bmatrix} \psi_0 \\ \mathbf{g}[\mathbf{y_0}, \mathbf{u_0}, \mathbf{p}, t_0] \\ \varsigma_0 \\ \mathbf{g}[\mathbf{y_1}, \mathbf{u_1}, \mathbf{p}, t_1] \\ \varsigma_1 \\ \vdots \\ \mathrm{g}[\mathbf{y}_{M-1}, \mathbf{u}_{M-1}, \mathbf{p}, t_{M-1} \\ \varsigma_{M-1} \\ \mathbf{g}[\mathbf{y_M}, \mathbf{u_M}, \mathbf{p}, t_M] \\ \psi_f \end{bmatrix}.$$

Tällainen epälineaarisen optimoinnin (NLP) ongelma on suuri sekä tilan että laskenta-ajan tarpeeltaan. NLP:n muuttujien ja rajoitteiden lukumäärä on $n \approx (n_y+n_u)MN$. Tällöin tyypillinen lentorata (7 tilaa, 2 ohjausta, 100 ruudukkopistettä/vaihe ja 5 vaihetta) tuottaa NLP:n jossa $n = 4500$. Hyvänä

puolena on tarvittavien matriisien eli Jacobin **G** ja Hessen **H** harvuus, jolloin niiden elementeistä saattaa alle 1 % olla erisuuria kuin nolla. Matriisien harvuuden onnistunut hyväksikäyttäminen algoritmissa onkin suorituskyvyn kannalta kriittinen tekijä.

Syöte: **x**
do for (jokaiselle vaiheelle) $k = 1, N$
  Alusta Vaihe k: tallenna ψ_0
  do for (jokaiselle verkon pisteelle) $j = 0, M - 1$
    Rajoitteen Arviointi: tallenna
      $\mathbf{g}[\mathbf{y}_j, \mathbf{u}_j, \mathbf{p}, t_j]$ ja ζ_j
  end do
  Lopeta Vaihe
    tallenna $\mathbf{g}[\mathbf{y}_M, \mathbf{u}_M, \mathbf{p}, t_M]$ ja ψ_f
end do
Lopeta Lentorata
Tulosta: $\mathbf{c}(\mathbf{x})$ ja $F(\mathbf{x})$

Algoritmi 5. Suora kollokaatiomenetelmä

3. Avaruustutkimuksen ongelmien optimointi tekoälymenetelmillä

John McCarthy loi käsitteen tekoälystä (Artificial Intelligence, AI) 1950-luvun puolivälissä. [Partridge, 1991]. Yksi suosittu tekoälyn määritelmä kuuluu seuraavasti: “AI is the study of how to make computers do things at which, at the moment, people are better” [Rich & Knight, 1991]. Vapaasti suomeksi käännettynä tekoälyn tutkimus kehittää tietokoneita suoriutumaan paremmin sellaisista tehtävistä, joissa ihmiset ovat vielä tietokoneita etevämpiä.

Optimoinnissa käytettäviä tekoälymenetelmiä ovat mm. evolutionääriset algoritmit, geneettiset algoritmit, muurahais- ja hiukkasparvioptimointi sekä tabu-etsintä. Näiden metodien perustana on optimointiongelman muuttujien arvojen valinnassa esiintyvä satunnaisuus [Betts, 1998].

Evoluutiolaskenta on tekoälyn ohjelmoinnin osa-alue, jonka avulla voidaan ratkaista monia jatkuvan ja kombinatorisen optimoinnin ongelmia. Evoluutiolaskennan algoritmeja voidaan luonnehtia globaaliksi optimoinniksi metaheuristiikan tai stokastisen optimoinnin menetelmien avulla. Optimoinnin kohteena on usein laskennallisesti vaativia ja raskaita mustan laatikon (black box) ongelmia, joihin on vaikea löytää optimaalista ratkaisua perinteisin menetelmin.

Jos mietitään optimointiongelmaa avaruudessa työskentelevän avaruusalusten parven kannalta, niin huomataan keskenään ristiriidassa olevia tavoitteita jo pelkästään ryhmän liikkumisessa. Tällaisia tavoitteita ovat mm. polttoaineen/energian kulutus, liikenopeus ja törmäysten välttäminen.

Evoluutiolaskennalle tyypillinen piirre on iteraatioon perustuva populaation kasvu tai kehittyminen eli biologiasta tuttu evoluutio. Algoritmien rakenne perustuu populaatioihin ja prosessissa käytetään opastettuja satunnaishakuja sekä rinnakkaisprosessointia.

Evoluutiolaskennan osa-alueisiin kuuluvat erilaiset evoluutioalgoritmitekniikat, kuten geneettiset algoritmit ja differentiaalievoluutio sekä parviälytekniikat, kuten muurahaiskoloniaoptimointi ja hiukkasparvioptimointi.

Edellä mainittujen lisäksi on olemassa myös muita populaatioon perustuvia metaheuristisia menetelmiä kuten meeminen algoritmi, jonka tavoitteena on yhdistää tehokkaasti paikallinen ja globaali etsintä.

3.1. Evoluutioalgoritmit

Evoluutioalgoritmit ovat populaatioon perustuvia metaheuristisia optimointialgoritmeja. Evoluutioalgoritmeissä on käytössä samoja käsitteitä kuin biologiasta tutussa evoluutioteoriassakin: populaatio, sukupolvet, geenivaraston muutos, (luonnon)valinta ja mutaatio. Evoluutioalgoritmi tuottaa ongelmalle useita ratkaisuehdotuksia, joista kelpoisuusfunktion avulla valitaan parhaat seuraavan suku-

polven populaatioon. Valintaprosessissa käytetään periytymistä, mutaatiota ja risteytystä. Kun evoluutioalgoritmi toistaa valintaprosessia useiden iteraatioiden ajan, lähestytään ratkaisuehdotuksissa optimaalista ratkaisua.

Dachwald ja Seboldt [2002] sekä Dachwald [2004] löysivät tutkimuksissaan neuroverkkojen ja evoluutioalgoritmien yhdistelmällä aikaisempaa optimaalisempia lentoratoja aurinkopurjetta käyttävälle avaruusalukselle, kun lentoratojen kohteina olivat mm. maapallon läheinen asteroidi $1996FG_3$ sekä Merkurius-planeetta.

Stracquadanio ja muut [2011] kehittivät evoluutiolaskentaan perustuvan black-box -algoritmin, SAGES (Self-Adaptive Gaussian Evolution Strategies), joka suoriutui hyvin monista Euroopan avaruusjärjestön (ESA) asettamista lentoradan optimoinnin esimerkkiongelmista. SAGES käyttää prosessissaan hyväksi kolmea muuta optimointialgoritmia; Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [Hansen et al., 2003], Differential Evolution (DE) [Price et al., 2006] sekä Divide RECTangle (DiRECT) [Jones et al., 1993].

”Patched conic approximation” on yleisesti käytetty menetelmä planeettojen välisen lentoradan laskemiseen, jonka ideana on jakaa lentorata yksittäisen taivaankappaleen painovoiman vaikutuspiirin mittaisiin osiin. Tällä tavalla tehtävä on huomattavasti yksinkertaisempi ratkaista kuin että kaikkien kappaleiden yhteisvaikutusta ratkaistaisiin yhtenä jatkuvana ongelmana. Luo ja muut [2011] yhdistivät tutkimuksessaan ”evolutionary patched model” tämän ja evoluutioalgoritmin saavuttaen hyvän tuloksen lentoradalle Maasta Marsiin ja sen lisäksi parannusta niin tehokkuuteen kuin laskutoimituksiinkin.

Uutena lähestymistapana Piotrowski ja muut [2014] hyödynsivät joukkouttamista (crowdsourcing) planeettojen välisen lentoradan määrittämiseksi. Heidän tutkimuksensa seuraa ihmisten menetelmiä pelin muodossa olevan optimointiongelman ratkaisemiseksi. Tutkimuksen tavoitteena on löytää uusia ja tehokkaampia optimointimenetelmiä.

3.1.1. Geneettiset algoritmit

John Holland [1975] esitti käsitteen geneettisistä algoritmeistä 1970-luvulla työskennellessään Michiganin yliopistossa. Holland kehitti geneettisiä algoritmeja biologian evoluutioteorian pohjalta. Geneettinen algoritmi toimii stokastisesti eli satunnaisesti. Evoluutioteorian periaatteella luodaan uusia ratkaisuvaihtoehtoja eli sukupolvia. Uuden sukupolven tuottamiseen vaikutetaan parhaiden yksilöiden stokastisella valinnalla sekä genomin muuntelulla. Algoritmi loppuu, kun on muodostettu ennalta sovittu maksimimäärä sukupolvia tai on saavutettu tyydyttävä kelpoisuusfunktion arvo.

Algoritmissa kromosomiksi koodatulle ongelmalle etsitään optimaalista ratkaisua, jota verrataan tavoitearvoon tai -funktioon. Geneettisillä algoritmeilla on neljä etua perinteisiin etsintäalgoritmeihin nähden [Buckles and Petry, 1992]:

- uusien ratkaisujen etsimisen ja aiemmin löydettyjen hyvien ratkaisujen hyödyntämisen välisen suhteen optimoiminen

- implisiittinen samanaikaisuus mahdollistaa laajan etsinnän ilman kaikkien arvojen suoraa testausta
- todennäköisyyteen perustuva satunnaissuus
- ratkaisuvaihtoehtojen samanaikainen käsittely vähentää paikallisen maksimin ja ”noisen” aiheuttamia ongelmia.

Kromosomien joukkoa kutsutaan populaatioksi, jota iteraatiolla kehitetään kohti tavoitetta. Evoluutioperiaatetta noudattaen parhailla kromosomeilla on valintavaiheessa suurempi todennäköisyys saada jälkikasvunsa mukaan seuraavaan populaatioon. Uusien kromosomien syntyyn vaikuttavat myös risteytys ja mutaatiot.

Kaksi suosittua metodia valinnan suorittamiseksi ovat Roulette Wheel ja deterministinen otanta [Buckles and Petry, 1992]. Roulette Wheel -valinnassa jokaiselle kromosomille lasketaan vanhemmaksi päätymisen todennäköisyys, jonka perusteella valinta suoritetaan satunnaistetusti. Deterministisessä otannassa edellä laskettua todennäköisyyttä sekä populaation kokoa käytetään määrittämään arvo, joka kertoo kuinka monta kertaa kyseinen kromosomi toimii vanhempana.

Risteytyksessä valitaan satunnaisesti kaksi kromosomia toimimaan vanhempina, joiden alleelien risteytyksen tuloksena syntyy uusi kromosomi. Mikäli uusi kromosomi läpäisee valinnan, niin se siirtyy uuden sukupolven populaatioon. Yleisenä tapana on esittää kromosomit binäärimuodossa, jolloin risteytys on helpompi suorittaa, mutta myös muita esitysmuotoja käytetään. Risteytys voidaan määrittää tapahtumaan satunnaisesti yhdestä kohdasta kromosomia (one-point crossover), esimerkiksi näin: vanhempi A (1 0 1 | 1 0 1) ja vanhempi B (0 0 1 | 1 0 0) tuottavat |:n kohdalta risteytettynä jälkeläiset lapsi A (1 0 1 1 0 0) ja lapsi B (0 0 1 1 0 1).

Syntyneillä uusilla kromosomeilla on ennalta määritelty todennäköisyys joutua mutaation kohteeksi. Mutaatiossa kromosomin osa, alleeli, saa uuden satunnaisen arvon. Mutaation avulla mahdollistetaan uuden geneettisen materiaalin tulo populaatioon ja geneettisen monipuolisuuden säilyminen.

Geneettinen algoritmi voi päättyä esimerkiksi ennalta sovitun iteraatiomäärän (sukupolvien) täyttyessä, jolloin paras vaihtoehto valikoituu ratkaisuksi. Eräs vaihtoehto on lopettaa algoritmi silloin, kun kaikki sukupolven kromosomit ovat samoja.

Xin-Sheng ja Li-Qun [2006] tutkivat ”nonholonomic”-järjestelmän liikkeen suunnittelun optimointia geneettisillä algoritmeilla aallokemuunnosapproksimaation (wavelet approximation) avulla. ”Holonomic”-järjestelmässä avaruusalus tai robotti kykenee kontrolloimaan liikettään kaikkien kuuden vapausasteen suhteen. ”Nonholonomic”-järjestelmän liikesuunnittelu on formuloitavissa optimiohjausongelmaksi. Aallokemuunnosmenetelmällä äärettömän ulottuvuuden ongelma muunnetaan äärellisen ulottuvuuden ongelmaksi. Ongelman ratkaisu geneettisellä algoritmilla tuottaa ”nonholonomic”-rajoitteet tyydyttävän lentoradan, joka demonstroidaan tehokkaaksi numeerisilla tuloksilla. Tutkimus osoitti samalla, että geneettistä algoritmia aallokemuunnosapproksimaatiolla voidaan käyttää myös ohjauksen syötteen saavuttamiseksi.

Xin-Sheng ja Li-Qun [2004] ovat tutkineet myös avaruusaluksen asennon kontrollointia geneettisellä algoritmilla, kun käytössä on vain kaksi vauhtipyörää (momentum wheel) normalin kolmen sijaan. Simulaatioissa geneettinen algoritmi osoittautui tehokkaaksi ja stabiiliksi.

Tripp ja Palmer [2009] tutkivat jatkuvan optimoinnin ongelmaa autonomisten nanosatelliittien toimiessa yhtenäisenä ryhmänä. Heidän tutkimuksensa kohdistuu populaation monimuotoisuuden säilyttämiseen uudenlaisilla korvausmenetelmillä, jotka osoittautuivat erittäin hyödyllisiksi vanhoihin korvausmenetelmiin verrattuna.

Kai ja Ming [2014] käyttävät geneettistä algoritmia muuttujien optimoimiseksi, kun satelliittia ohjataan uudelle kiertoradalle. Abdelkhalik ja Mortari [2007] tutkivat satelliitin ohjaamista uudelle kiertoradalle vain kahden ohjausimpulssin avulla. Heidän algoritminsa takaa ratkaisun, vaikka geneettinen algoritmi ei suppenisi globaaliin optimiin.

Danoy ja muut [2012] ovat kehittäneet geneettiseen algoritmiin perustuen uudeen parhaan tunnetun ratkaisun Cassini 2 -luotaimen lentoradan optimoinnin ongelmaan. Geneettisen algoritmin ja Nelderin-Meadin simplex-metodin yhteistyöllä HH-cGA-algoritmi tarvitsi ongelman ratkaisemiseen 25 kertaa vähemmän laskutoimituksia kuin verrokkialgoritmi. Bo ja Feng [2011] puolestaan tutkivat konstellaation satelliittien tankkaamista muodostelmalentoon perustuen yhden ja kahden tankkaajan taktiikalla, joista jälkimmäinen osoittautui sitä tehokkaammaksi, mitä enemmän satelliitteja konstellaatio sisälsi ja mitä vähemmän tehtävälle annettiin aikaa.

3.1.2. Differentiaalievoluutio

Differentiaalievoluutio (DE) ei vaadi optimointiongelman derivoituvuutta niin kuin perinteiset optimointimenetelmät, kuten ”gradient descent” ja ”quasi-newton”. DE on rinnakkain toimiva suoran etsinnän metodi, jossa optimointiongelman parametreistä muodostetaan satunnaisesti parametrivektoreita. Yhdessä nämä vektorit muodostavat algoritmin populaation. Menetelmässä hyödynnetään niin mutaatiota, risteytystä kuin valintaakin. Mutaatiossa syntyy uusi vektori siten, että DE generoi uuden parametrivektorin lisäämällä populaation kahden vektorin välisen painotetun eron kolmanteen vektoriin. Mutaatiossa syntyneen vektorin parametrit puolestaan sekoitetaan ennalta valitun kohdevektorin kanssa risteytyksessä. Mikäli näin syntynyt uusi vektori on parempi kuin edellä mainittu kohdevektori, niin kohdevektori korvataan tällä uudella vektorilla seuraavan sukupolven populaatiossa. Vimeinen vaihe toimii siis valintana. Populaation jokainen vektori joutuu kertaalleen kohdevektoriksi [Storn and Price, 1997].

Differentiaalievoluutio on stokastinen optimointimenetelmä, joka soveltuu hyvin planeettojen välisten lentoratojen suunnitteluun. Olds ja muut [2007] huomasivat tutkimuksessaan differentiaalievoluution herkkyyden algoritmin rutiinin säätöparametrien valinnalle. Rutiinin oikealla hienosäädöllä globaalin optimin ratkaiseminen oli huomattavasti nopeampaa.

Uusilla mutaatio- ja risteytysmenetelmillä Islam ja muut [2012] puolestaan paransivat tilastollisesti merkittävästi differentiaalievoluutioon perustuvaa globaalia optimointia, kun laskettiin Messenger ja Cassini 2 -luotainten lentoratoja.

3.2. Parviäly

Parviälytekniikat ovat populaatioihin perustuvia stokastisia menetelmiä kombinatoristen optimointiongelmien ratkaisemiseksi [Hinchey et al., 2005]. Ajatuksen parviälystä esittivät ensimmäisinä Beni ja Wang [1989] soluautomaattien tutkimuksessaan. Parviälyssä suhteellisen yksinkertaiset yksilöt eli agentit luovat globaaleja käyttäytymismalleja toimiessaan paikallisessa vuorovaikutuksessa toistensa ja ympäristönsä kanssa ilman minkäänlaista keskitettyä johtamista tai hallintaa. Luonnossa hyviä esimerkkejä toimivasta parviälystä ovat muurahaiskoloniat, heinäsirkkaparvet, mehiläispesät sekä monien nisäkkäiden käyttäytyminen parvissa ja laumoissa.

Parviälylle on kehitetty monia erilaisia algoritmeja, joista tunnetuimpia ovat muurahaiskoloniaja hiukkasparvioptimointi.

3.2.1. Muurahaiskoloniaoptimointi

Muurahaiskoloniaoptimointi (Ant colony optimization, ACO) on populaatioon perustuva metaheuristiikka, jota voidaan hyödyntää etsittäessä approksimoitua ratkaisua vaikeisiin optimointiongelmiin [Dorigo, 2007]. Tässä menetelmässä siis joukko agentteja etsii hyviä ratkaisuja tiettyyn optimointiongelmaan. Agentteja kutsutaan myös muurahaisiksi (ants).

Menetelmän alussa ratkaistava optimointiongelma on muutettava parhaan polun etsimisongelmaksi painotetussa graafissa eli verkossa. Seuraavassa vaiheessa agentit alkavat edetä verkossa samalla tuottaen enenevässä määrin uusia ratkaisuja. Ratkaisun rakentamisprosessi on luonteeltaan stokastinen ja painotettu feromonisen mallin mukaisesti. Feromonisella mallilla tarkoitetaan graafin osiin (solmut ja kaaret) liittyvien parametrien joukkoa, joiden arvoja muurahaiset muuttavat algoritmin suorittamisen aikana [Dorigo, 2007]. Feromonin määrä solmussa tai kaaressa opastaa muurahaista oikeaan suuntaan parhaan polun etsinnässä. Algoritmissa 6 esitellään muurahaiskoloniaoptimoinnin metaheuristiikkaa Dorigon [2007] näkemyksen mukaisesti.

```
Aseta parametrit, alusta feromonien polut
TOIMINTOJEN_AJOITUS
  KonstruoiMuurahaistenRatkaisut
  DaemonTehtävät          {valinnainen}
  PäivitäFeromonit
LOPETA_TOIMINTOJEN_AJOITUS
```

Algoritmi 6. Muurahaiskoloniaoptimointi

Muurahaiskoloniaoptimoinnin metaheuristiikka koostuu alustusvaiheesta ja kolmesta algoritmisesta komponentista, joiden aktivoitumista säätelee TOIMINTOJEN_AJOITUS rakenne. Tätä rakennetta toistetaan siihen asti, että lopetuskriteeri on saavutettu. Kriteerinä on usein joko iteraatioiden määrä tai käytetty laskenta-aika. TOIMINTOJEN_AJOITUS ei automaatisesti määritä näiden kolmen algoritmisen komponentin suorittamista ja synkronointia, vaan se on täysin ohjelman toteuttajan määritettävissä. Usein nämä komponentit suoritetaan silmukassa, jossa a) konstruoidaan kaikkien muurahaisten kaikki ratkaisut, b) mahdollisesti parannetaan ratkaisuja paikallisen etsintäalgoritmin avulla ja c) päivitään feromonit [Dorigo, 2007].

Dorigon [2007] mukaan kolme menestyneintä ACO-menetelmää ovat Ant System (AS) [Dorigo, 1992; Dorigo et al., 1991; 1996], Ant Colony System (ACS) [Dorigo and Gambardella, 1997] ja MAX-MIN Ant System (MMAS) [Stützle and Hoos, 2000]. Suominen [2013] vertaili työssään AS- ja ACS-menetelmiä klassisen kauppamatkustajan ongelman ratkaisemisessa. Hänen tutkimuksessaan ACS osoittautui nopeammaksi menetelmäksi kaikissa kolmessa eri tutkimustapauksessa, mutta AS löysi parhaimmillaan lyhyempiä ratkaisuja kuin ACS.

Iacopino ja muut [2014] keskittyivät tutkimuksessaan maapallon tilaa tarkkailevan avaruusalusten konstellaation yhteistoiminnan suunnitteluun. Uutena ideana tällaisen tehtävän suunnittelemiseksi oli luoda itseorganisoiva multiagenttiarkkitehtuuri, joka kykenee mukautumaan tehtävän aikaisiin muutoksiin sekä synkronoimaan satelliittien suunnitelmat siten, että vältytään päällekkäisiltä toiminnoilta. Heidän ACO-menetelmään perustuva ratkaisunsa osoittautui hyödylliseksi mukautuvuuden ja koordinoinnin suhteen verrattuna standardeihin optimointialgoritmeihin kuten geneettiseen algoritmiin.

Schlueter [2012] tutki epälineaariseen sekalukuoptimointiin (MINLP) perustuvaa tekniikkaa kohdistettuna käytännön ongelmiin varsinkin avaruustutkimukseen liittyvissä sovelluksissa. Hänen algoritmissaan oli kaksi pääkomponenttia, joista toinen oli laajennus ACO-metaheuristiikkaan ja toinen rajoitteiden hallintaan (Oracle Penalty Method). Algoritmista kehitetty MIDACO-ohjelma osoittautui kilpailukykyiseksi perinteisiin MINLP-algoritmeihin verrattuna. Lisäksi algoritmi kykeni ratkaisemaan haastavia avaruussovellusten ongelmia, joita hänen työssään ensi kertaa testattiin sekalukuongelmina. Testattuja avaruusongelmia olivat mm. Satellite Constellation Optimization, Global Trajectory Optimization Problems sekä Interplanetary Space Mission Design.

Euroopan avaruusjärjestön (ESA) organisoiman GTOP-kilpailun (Global Trajectory Optimization Problems) seitsemästä tehtävästä viiteen löytyi MIDACO:n avulla paras ratkaisu. Ratkaisuista kaksi olivat lisäksi ennestään tuntemattomia. Laskentaan vaadittu aika vaihteli 39 minuutista aina 50 päivään asti (PC, 1,66 GHz, 1 GB RAM).

Ceriotti ja Massimiliano [2010] esittivät tutkimuksessaan Ant System -pohjaisen ratkaisun automoidulle lentoradan suunnittelulle, kun lentoradan aikana käytetään hyväksi planeettojen painovoimaa. Mahdollisten lentoratojen määrä kasvaa eksponentiaalisesti suhteessa vierailtavien planeettojen määrään. Tutkimuksessa ehdotettu "T-ACO"-algoritmi (algoritmi 7) välttää tarpeen laskea kaikki vaihtoehdot ja tuottaa hyviä tuloksia pienellä laskentatarpeella. Ant Systeemiin perustuen jokainen muurahainen (ant) tekee jokaisen planeetan kohdalla päätöksensä siinä sijaitsevien tabu- ja toteuttamiskelpoisuustietojen perusteella. Algoritmi on käytännön testeissä osoittautunut kilpailukykyiseksi tunnettuihin geneettisiin ja hiukkasparvialgoritmeihin verrattuna.

Aseta muurahaisten lukumäärä yhtä kuin m
for kaikille $k = 1, \ldots, m$ do
  Generoi planeettojen jono
  Generoi siirtymien tyypit
  if s ei hylätty then
    $S \leftarrow S \cup \{s\}$
  end if
end for
Arvioi kaikki S:n ratkaisut
Päivitä feasible lista ja tabu lista
Lopetus jollei $n_{iter} > n_{iter,\,max} \vee n_{eval} > n_{eval,\,max}$, GoTo Alku

Algoritmi 7. T-ACO algoritmi

Muurahaisten jättämät feromonijäljet ovat inspiroineet tiedonvälitystä myös Trippin ja Palmerin [2010] tutkimassa Stigmergy-menetelmässä, jossa satelliittijoukon toiminnan koordinointi perustuu avaruuden kyseessä ollessa digitaaliseen ympäristöön jätettyihin viesteihin. Kyseinen menetelmä vähentää satelliittien välisen kommunikoinnin tarvetta.

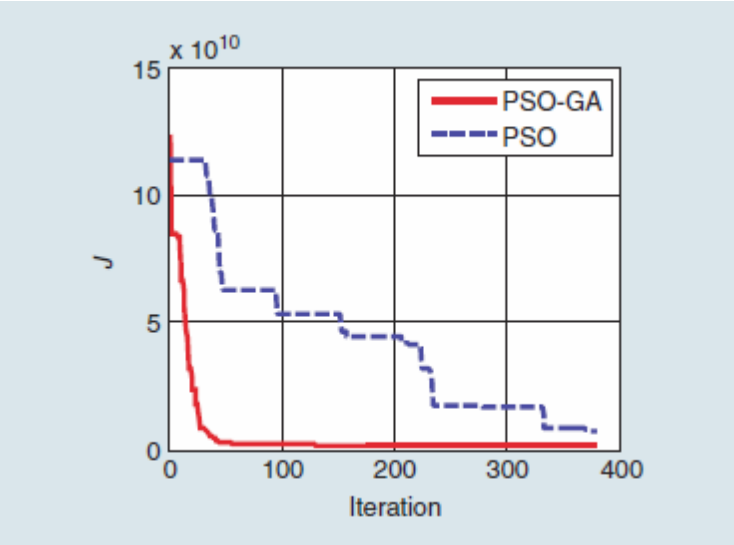
Työssä tutkittiin mm. kahta tärkeää osa-tavoitetta eli tehtävien kaksinkertaisen suorittamisen minimoimista sekä korkean prioriteetin tehtävien nopeaa suorittamista. Kävi ilmi että, Stigmergy ei kykene maksimoimaan molempia tavoitteita samanaikaisesti. Tutkijat olettavat, että menetelmä toimii vielä paremmin, kun toimitaan laajemmassa ympäristössä ja monimutkaisempien ongelmien parissa, kuin mitä he nyt simulaatioissa testattiin.

3.2.2. Hiukkasparvioptimointi

Idean hiukkasparvioptimoinnista esittivät Eberhart ja Kennedy [1995]. Se on populaatioon perustuva stokastinen lähestymistapa jatkuvien ja diskreettien optimointiongelmien ratkaisemiseksi. Hiukkasparvioptimoinnissa ohjelma-agentit eli hiukkaset kulkevat optimointiongelman etsintäavaruudessa. Hiukkasen sijanti edustaa kyseisen ongelman ratkaisuehdostusta. Jokainen hiukkanen etsii parempaa sijaintia etsintäavaruudessa muuttamalla nopeuttaan. Nopeuden muutosta sääntelevät säännöt ovat alun perin peräisin lintuparvien käyttäytymismalleista. Algoritmin pääsilmukassa suoritetaan hiukkasten nopeuksien ja sijaintien päivitys iterativiisesti kunnes lopetusehto on saavutettu [Dorigo et al., 2008].

Simoes ja muut [2012] kehittivät tutkimuksessaan hybridisen PSO-Tabu -algoritmin, joka reaaliajassa etsii planeetalta sopivaa laskeutumispaikan. Hiukkasparvioptimoinnin tehokkuus globaalin eksploratiivisen etsinnän saralla yhdistettynä paikalliseen tabu-etsintään tuotti poikkeuksellisen hyviä tuloksia. IPSIS (Intelligent Planetary SIte Selection) Piloting-funktion ympärille rakennettu uusi HDA (Hazard Detection and Avoidance) -arkkitehtuuri nopeutti ratkaisua 30–44-kertaiseksi verrattuna IVN+ (Integrated Vision and Navigation) -ohjelmistoon testin erilaisilla laitteistoalustoilla.

Duan ja muut [2013] tutkivat muodostelmassa lentävien miehittämättömien lentokoneiden muodestelman uudelleenkonfiguroimisen ongelmaa hiukkasparvioptimoinnin ja geneettisen algoritmin yhdistelmän avulla. Kuten kuvasta 2 voimme havaita heidän kehittämänsä “Hybrid Particle Swarm Optimization and Genetic Algorithm” (HPSOGA) algoritmi oli huomattavasti perinteistää PSO-ratkaisua parempi monimutkaisissa ympäristöissä.



Kuva 2. HPSOGA vs. PSO

3.3. Meeminen algoritmi

Ajatuksen meemisestä algoritmista (MA) esitti ensimmäisenä Moscato [1989]. Hänen mielestään MA pyrkii matkimaan kulttuurista evoluutiota. Algoritmin toimintaperiaatteena on populaatioon perustuvan globaalin etsinnän ja sen yksilöiden suorittama heurstinen paikallisen etsinnän yhteistyö.

Meemistä evoluutioalgoritmia on ehdottu ratkaisuksi robottiparven liikkeen ja muodostelman ohjaukseen [Lin et al., 2012]. Ehdotettu algoritmi koostuu globaalista polun suunnittelijasta sekä paikallisesta liikkeen suunnitelijasta ja keskittyy polun turvallisuuteen. Globaali polun suunnittelija laskee lentoratoja vapaassa avaruudessa olevassa Voronoin diagrammissa. MA suunnittelee ennalta määrätylle alkuperäispopulaatiolle sarjan konfiguraatioita edellisessä vaiheessa määritellyn polun perusteella. Tämän jälkeen MA etsii paikallisesti yksittäisille roboteille parempaa hyvyysfunktion arvoa välietappien saavuttamiseksi. Optimaalisen konfiguraation löydyttyä parhaat kromosomit varataan seuraavan sukupolven alkupopulaatioksi. Koska tämä MA käyttää ei-satunnaista alkupopulaatioita ja paikallista etsintää, on se tehokkaampi kuin perinteinen geneettinen algoritmi ja löytää myös nopeamman polun.

Polun turvallisuuden painottamiseksi käytetään hierarkkista polunetsintäalgoritmia. Globaali polunetsintäalgoritmi etsii sellaista reittiä, jota pitkin robottiparven keskipiste etenee. Liikkeen suunnittelija puolestaan on potentiaalikenttään perustuva geneettinen algoritmi. Näillä potentiaaleilla robotit välttävät törmäykset ja pysyvät määrätyllä etäisyydellä toisistaan. Lasketun polun turvallisuuteen kuuluu sekä turvallinen globaali polku että turvallinen liike paikallisesti. Globaali polku on optimaalinen turvallisuuden suhteen, koska se on maksimaalisesti etäällä esteistä. Paikallinen liike puolestaan on optimaalisesti turvallinen, koska se huomioi etäisyyden seuraavaan välietappiin, etäisyyden esteisiin ja robottien yhdistettävyyden. Yleisesti ottaen Lyaponov-pohjaiset geneettiset algoritmit perustuvat stabiliteettiin ja vakauteen, jossa kromosomin parhaan hyvyysarvon paraneminen on olematonta [Pereira et al., 2008]. Nyt kyseessä olevassa tapauksessa robottiparven optimaalinen konfigurointi saavutetaan evoluution tultua stabiiliin tilaan.

Koska globaalin ja paikallisen polun etsinnät ovat kytköksissä toisiinsa, tulee monista algoritmeista monimutkaisia ja ne saattavat pysähtyä paikalliseen minimiin. Globaalin polun etsinnän laskennellallisuutta yksinkertaistaa se, että ei suoriteta koko Voronoin diagrammin laskentaa ja polun etsimistä kerralla, vaan jaetaan se pienempiin osadiagrammeihin ja -polkuihin.

Algoritmi 8 kuvaa potentiaaliin perustuvaa meemistä algoritmia Lin ja muiden [2012] työssä.

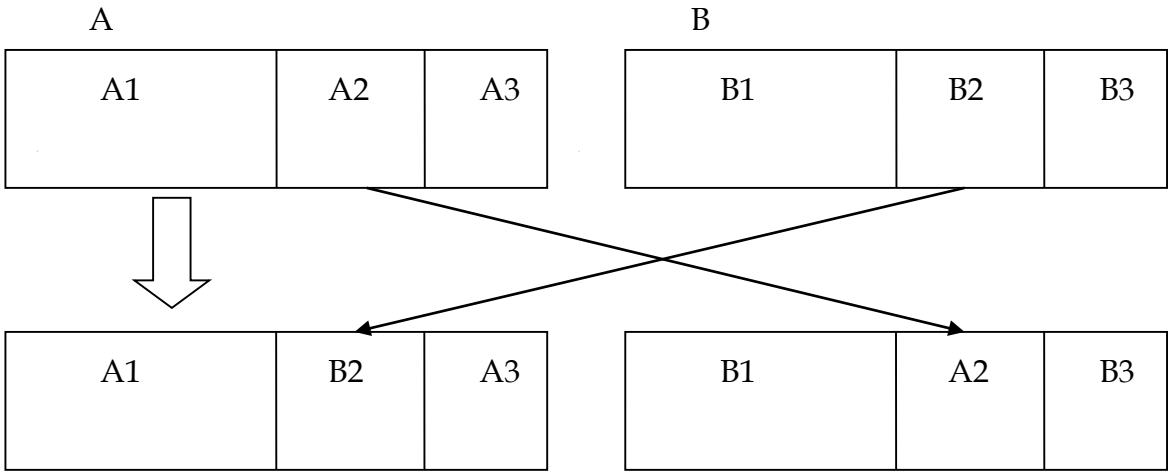
```
Aloita
i = 1; /* Alusta ensimmäinen välitavoite */
t = 0; /* Alusta evolutionääriset sukupolvet */
Generoi satunnaisesti ensimmäinen populaatio P_i ( t );
kuntoisuus (P_i(t));
    toista kunnes (saavuta lopullinen tavoite q_n) Do
    P_{i+1}(t)= P_i(t);
        toista kunnes (saavuta välitavoite q_i)
        Do
            valitse P_i(t+l) from P_i(t);
            risteytys (P_i(t+l));
            mutaatio (P_i(t+l));
            kuntoisuus (P_i(t+l));
            toteuta Hienosäätönä Paikallinen Etsintä to P_i(t+l)
            t = t +1;
        end
    i= i +1;
end
Lopeta
```

Algoritmi 8. Meeminen algoritmi

Algoritmin työvaiheita ovat populaation alustus, luonnollinen valinta, risteytys, mutaatio, kuntoisuus ja paikallinen etsintä. Ensimmäinen populaatio luodaan satunnaisesti, mutta sitä seuraavat sukupolvet ovat osaksi periytyneitä ja osaksi satunnaisia. Evoluutioon kuluvaa aikaa pienentää alustettavien populaatioiden eugenisuus.

Luonnollisessa valinnassa valitaan populaatiosta parhaita kromosomeja sisällytettäväksi seuraavaan sukupolveen. Tässä kohdassa kromosomi voi joutua risteytyksen tai mutaation kohteeksi, jolloin siitä syntyvä jälkeläinen pääsee uuteen sukupolveen. Parhaita kromosomeja valittaessa käytetään hyväksi kuntoisuuden arviointiin kehitettyä hyvyysfunktiota. Kuntoisuutta voidaan arvioida erilaisilla objektiivisilla menetelmillä. Tässä tapauksessa valitaan yksinkertaisesti ensimmäiset 10 % parhaista arvoista ja loput valitaan satunnaisesti.

Ristetystä ja mutaatiota käytetään, koska pelkällä valinnalla ei populaatioon luoda uusia yksilöitä. Risteytyksessä valitaan kaksi kromosomia vanhemmiksi ja vaihtamalla niiden geenejä keskenään saadaan kaksi uutta jälkeläistä. Tässä algoritmissa kromosomin *i.* geeni kuvaa robotin *i* sijaintia, joten risteytyksessä vaihdetaan samoin sijoittuneita kromosomiparien geenejä (kuva 3). Muutaatiossa puolestaan mielivaltainen osa geeneistä vaihdetaan todennäköisyyden perusteella. Mutaation tarkoituksena on ylläpitää geneettistä monipuolisuutta ja estää ajautumista paikalliseen minimiin.



Kuva 3. Risteytys

Hyvyysfunktiolla tarkoitetaan tässä tapauksessa robotin etäisyyttä tavoitteestaan. Osana tätä funktiota vaaditaan myös törmäysten välttämistä eli optimaalinen kromosomi on mahdollisimman lähellä maalia eikä törmää esteisiin. Parven yhtenäisyyden säilyttämiseksi robotin on pidettävä määrätty etäisyys parven keskipisteeseen sekä muihin robotteihin.

Kyseessä olevan memeettisen algoritmin paikallisessa etsinnässä käytetään hienosäätöominaisuutta optimaalisen ratkaisun löytämiseksi. Perinteisistä geneettisistä algoritmeista tämä ominaisuus puuttuu. Hienosäätöä käytetään parven keskipisteen parantamiseksi. Esitetyn algoritmin muokkaaminen kolmiuloitteisessa avaruudessa toimivaksi on myös mahdollista ilman merkittävää lisätyötä.

Lin ja muiden [2012] simulaatioiden tulosten perusteella ehdotettu algoritmi toimii hyvin polun ”smoothnessin” ja laskennan tehokkuuden suhteen verrattaessa perinteiseen geneettiseen algoritmiin kolmen ja kahdeksan robotin parvessa. Samanlaiseen tulokseen päätyivät myös Kloetzer ja Belta [2007] 30 robotin muodostaman parven tapauksessa.

3.4. Sumea logiikka

Sumeassa logiikassa totuusarvo on perinteisestä logiikasta poiketen reaalinen luku suljetulla välillä nollasta ykköseen. Sumeaan logiikkaan muistuttaa paljolti ihmisen tapaa toimia ja tehdä päätöksiä. Näin ollen se soveltuukin erittäin hyvin tekoälyn tutkimiseen. Avaruustutkimuksessa sumen logiikan käyttöä on vähentänyt luottamuspula menetelmän vakauteen.

Karr ja Freeman [1997] paransivat sumeaan logiikkaan perustuvaa kahden avaruusaluksen kohtaamista geneettisen algoritmin avulla. He myös totesivat geneettisten algoritmien mahdollistavan tehokkaan ja strukturoidun vaihtoehdon sumean logiikan järjestelmien suunnittelemiselle.

Giron-Sierran ja Ortegan [2002] työssä kartoitettiin sumean logiikan ohjaimen (Fuzzy Logic Controller) vakautta avaruuden ohjelmistoissa. Sumeaan logiikan ohjainta käytetään mm. seuraavissa avaruusalusten ja satelliittien toiminnoissa: asennon hallinta, kohtaamiset ja paluu ilmakehään.

Zou ja Kumar [2011] tutkivat avaruusaluksen asennon hallitsemista sumean logiikan avulla erilaisissa ongelmatilanteissa, kuten aktuaattorin vikatilassa. Sumeaan logiikkaan ja ”backstepping”-tekniikkaan perustuvalla sopeutuvalla ohjaimella pystyttiin simulaatioissa takaamaan asennon hallinta useissa eri ongelmatilanteissa säätämällä toimintaan kuluvaa aikaa sekä ohjauksen vääntömomenttia.

4. Moderneja sovelluksia

Avaruustutkimuksen suunnittelu esim. lentoratojen suhteen on aina vaatinut paljon laskennallista työtä ja kun tehtävät käyvät kunnianhimoisemmiksi ja monimutkaisemmiksi ei tietokoneissakaan enää riitä laskentateho tulosten saavuttamiseksi järkevässä ajassa. Helpotusta asiaan on haettu ja löydetty tekoälyn tutkimuksesta ja evoluutiolaskennasta.

Evolutionääriset ja geneettiset algoritmit tuovat uusia ja parempia ratkaisuja, kun pitää optimoida lentoratoja samanaikaisesti esim. ajankäytön, matkan pituuden ja turvallisuuden suhteen.

Seuraavassa käydään läpi muutamia avaruustutkimukseen liittyviä esimerkkitapauksia, joiden toteuttamisessa käytetään hyväksi moderneja tekoälymenetelmiä ja näin yritetään lisätä projekteihin sekä autonomiaa että älykkyyttä.

4.1. Autonomous Nano Technology Swarm (ANTS)

Amerikan Yhdysvaltojen avaruustutkimuslaitos, NASA, on kiinnostunut autonomisista järjestelmistä avaruudessa tapahtuvien operaatioiden onnistumistodennäköisyyden parantamiseksi sekä kulujen karsimiseksi [Truszkowski and Hallock, 1999]. NASA on huomannut ihmisten ja robottien välisen yhteistyön kehittämisen tärkeyden turvallisten, tehokkaiden ja taloudellisesti kannattavien tutkimustehtävien kannalta.

NASA on kehittänyt ANTS-arkkitehtuuria (Autonomous Nano Technology Swarm) avaruudessa tapahtuvien autonomisten tutkimusprojektien toteuttamiseksi. ANTS-arkkitehtuuria on suunniteltu käytettäväksi mm. seuraavissa NASAn projekteissa: LARA (Lander Amorphous Rover Antenna), PAM (Prospecting Asteroid Mission) ja SARA (Saturn Autonomous Ring Array). Myös avaruusteleskooppi Hubblen pelastamiseksi on tehty suunnitelma ANTSia hyväksi käyttäen. ANTSin teknologia rakentuu uusimmalle kehitykselle robotiikassa, tekoälyssä ja materiaaleissa; tavoitteena on minimoida kulut ja maksimoida tehokkuus avaruusoperaatioissa.

ANTS-arkkitehtuurin perustuvien projektien avulla on mahdollista toteuttaa ihmisten ja robottien yhteistyö avaruustutkimuksessa, jonka päämääränä on laajentaa ymmärtämystämme aurinkokunnastamme ja hyödyntää avaruudessa olevia resursseja. ANTS oli ensimmäisen kerran esillä asteroidivyöhykkeen tutkimusta pohtivassa esitelmässä [Curtis et al., 2000].

ANTS-arkkitehtuuria on kehitetty perustuen sosiaalisten hyönteisparvien toimintaan [Curtis et al., 2000; Curtis et al., 2003a]. Luonnossa esiintyvät muurahaiset ovat yksi menestyneimmistä ja sopeutumiskykyisimmistä eläinlajeista. Muurahaisten menestyksen taustalla on mm. onnistunut työnjako, yhteistyö ja muurahaisten lukumäärä [Curtis et al., 2003b]. Lukumäärä lisää redundanssin kautta luotettavuutta monissa parven tehtävissä. Yksittäiset muurahaiset ovat jo sinänsä erittäin autonomisia ja pystyviä yksiköitä, jotka kuitenkin ovat riippuvaisia yhteisönsä muista jäsenistä, jotta

selviytymisen ja lisääntymisen kannalta tärkeät tehtävät suoritetaan onnistuneesti. Yhteen asiaan erikoistuneella muurahaisella on paljon suurempi todennäköisyys onnistua tehtävässään kuin jos tilalla olisi yleispätevä muurahainen. Muurahaiset ja niiden parvet hyötyvät menestyksestään ja muurahaisyhteiskunta kehittyy kilpailuetunsa turvin koko ajan tehokkaammaksi. ANTS pyrkii jossain määrin luomaan yhtä sitkeän, menestyvän ja mukautuvan järjestelmän kuin muurahaisyhteiskunnat. Luonnossa elävät muurahaiset kykenevät suhteellisen sofistiseen käyttäytymiseen suhteellisen yksinkertaisilla neurorakenteilla ja viestinnän mekanismeilla. ANTS-arkkitehtuuri pyrkii simuloimaan sekä yksilöiden että parven kykyjä.

ANTSin on tarkoitus olla autonominen, heuristinen, skaalautuva, kestävä ja erittäin hajautettu järjestelmäarkkitehtuuri [Curtis et al., 2003]. Arkkitehtuurin tärkeitä piirteitä ovat analogia sosiaalisten hyönteisparvien kanssa, itsenäiset ja erikoistuneet yksilöt sekä järjestelmän monitasoinen älykkyys ja autonominen toiminta. ANTS koostuu parvesta autonomisia ANT-yksilöitä, jotka toimivat itsenäisinä yksikköinä ilman keskinäisiä ristiriitoja. Parven yksilöt muodostavat hierarkkisen älykkyyden yhdessä muiden yksilöiden kanssa. Parvella on sosiaalinen rakenne, jossa yksilöt toimivat yhdessä saavuttaakseen tehtävän päämäärät. Parvi voidaan tarvittaessa jakaa pienempiin toiminnallisiin osiin, jotka kykenevät itsenäiseen työskentelyyn. Ryhmiä voidaan muokata eri tarkoituksia varten, jolloin osa ryhmistä voi olla pysyviä ja loput yksilöt ovat tarpeen mukaan joko osa kokonaisparvea tai omia ryhmiään. ANTS-parvi voi sisältää satoja pieniä avaruusaluksia tai satelliitteja, jolloin se on myös erittäin vikasietoinen.

ANTS-arkkitehtuuri edellyttää erittäin autonomisia avaruusaluksia, jotka ovat erikoistuneet tiettyyn tehtävään kuten datan keräämiseen, kommunikaatioon tai tehtävän johtamiseen. Erikoistuminen parantaa rajallisten resurssien käyttöä, mutta vaatii yhteistyötä parven jäseniltä, jotta kokonaistavoitteet toteutuvat. Kaikkien järjestelmän avaruusalusten pitää suoriutua myös perustavanlaatuisista selviytymistehtävistä. Näihin kuuluvia toimintoja ovat esimerkiksi navigointi ja törmäysten välttäminen. Lisäksi ANTSin yksilöiden on oltava kestäviä ja riittävän älykkäitä toimiakseen myös ongelmatilanteissa.

Koska avaruusalukset ovat eksperttejä omalla erikoisalueellaan, voidaan niiden fyysinen ulkomuoto, toiminnot ja lentoradat optimoida aluksen tehtävän mukaisesti. Näillä keinoilla parannetaan huomattavasti projektien tieteellistä arvoa, yksinkertaistetaan datan analysointia ja päästään parempaan lopputulokseen.

Redundanssi avaruusalusten tasolla parantaa luotettavuutta ja virhetoleranssia. Vaikka yksittäinen avaruusalus on optimoitu tiettyyn tehtävään, on myös mahdollista, että se yrittää sellaisia riskialttiita liikkeitä, joihin koko tehtävää yksin suorittava avaruusalus ei missään tapauksessa ryhtyisi. ANTS-arkkitehtuuria voidaan soveltaa laaja-alaisesti erilaisissa avaruusalusten muodostelmissa.

Tällä hetkellä mahdollisimman reaaliaikainen valvonta ja ohjaus ovat elintärkeitä avaruustehtävien onnistumisille. Autonomian kehittyessä tarve ihmisten suorittamalle valvonnalle ja ohjaukselle voidaan minimoida, jolloin kustannukset laskevat ja tuottavuus paranee.

Tämän hetkiset suunnittelumallit ja avaruusalusarkkitehtuurit on helpointa luokitella yleistetyiksi ainakin tieteellisten tutkimustehtävien osalta. Yleistämistä tapahtuu, koska tämänhetkiset tehtävät vaativat avaruusalukselta monia erilaisia tieteellisiä instrumenttejä. Mitä enemmän tutkimusinstrumenttejä ja -tavoitteita avaruusaluksella on, sitä useammin toiminnot ovat ristiriidassa sekä keskenään että avaruusaluksen turvallisuuteen liittyvien tavoitteiden kanssa, jolloin joudutaan tekemään kompromisseja tutkimusten suorittamisen suhteen.

ANTS-arkkitehtuuri käyttää kehittyvää hierarkkista neurojärjestelmää, jossa yksittäinen avaruusalus edustaa korkeimman tason solmua (node). Järjestelmä hyödyntää parviälyä ja solmujen yhteistyötä järjestelmän kaikilla tasoilla. Korkeimmilla tasoilla tämä näkyy kollektiivisena toimintana esimerkiksi yksittäisten avaruusalusten muodostamassa suuressa muodostelmassa. ANTSin arkkitehtuuri on suunniteltu monimutkaisten järjestelmien täysin autonomiseen toimintaan. Edellä mainitun PAM-projektin vaatimukset pakottavat miettimään lähestymistapoja, jotka ovat erittäin hajautettuja ja täysin autonomisia, mutta silti luotettavia ja tarpeeksi yhtenäisiä suoriutuakseen tehtävän päämääristä tehokkaasti.

ANTS-arkkitehtuuri voidaan jakaa alemman ja ylemmän tason toimintoihin. Alemman tason toimintoihin kuuluu perustoimintoja kuten huolehtiminen aluksen kunnosta ja turvallisuudesta. Nämä toiminnot vaativat korkeaa autonomiaa, jotta yksilön ja parven toimintakyky on turvattu. Alemman tason toiminnot vastaavat paljolti järjestelmän luotettavuudesta ja käytettävyydestä.

Ylemmän tason toimintoihin kuuluvat toimintojen suunnittelu, päätösten tekeminen, datan analysointi, sosiaalinen vuorovaikutus, parven yhteistyö sekä vikojen diagnosointi ja korjaaminen. Näitä ongelmia varten kehitetään heuristisia järjestelmiä sekä erilaisia tekoälytekniikoita. Esimerkiksi geneettisillä algoritmeilla voidaan kehittää ja luoda uusia strategioita tai eliminoida tuottamattomia strategioita. Tekoäly on oleellinen osa kaikkia ANTSin järjestelmiä. ANTS-arkkitehtuuri vähentää järjestelmien kompleksisuutta tarjoamalla alemman tason autonomisia toimintoja ylemmän tason tekoälyn kontrolloimana.

Koska yksi järjestelmän vaatimuksista on tieteen tekeminen paikallisesti, tekee ANTS tehtävän käynnistyttyä itsenäisiä päätöksiä mm. tieteellisten instrumenttien käytöstä, tutkimuskohteiden valinnasta ja navigoinnista. Tietoa päätöksenteon pohjaksi on annettu etukäteen, mutta tieto myös lisääntyy tehtävän aikana. Kun tehtävä edistyy ja tulee uutta informaatiota, voivat tehtävän prioriteetit muuttua ja toiminnalliset metodit mukautua uuteen tilanteeseen. Tämän pitäisi tapahtua ilman ihmisten puuttumista asioihin ja se voidaan saavuttaa tekoälyn ja koneoppimisen avulla kehittyvällä järjestelmällä.

ANTSin kannalta kriittinen lähiajan tavoite on perustavanlaatuisen alemman tason autonomian sekä ylemmän tason autonomisen ohjausjärjestelmän kehittäminen. Tutkimustyön alla on myös tehtävien itsenäinen suorittaminen, reittien suunnittelu, etsintämenetelmien optimointi, kommunikaatio, tieteellisen datan organisointi sekä resursseista huolehtimen.

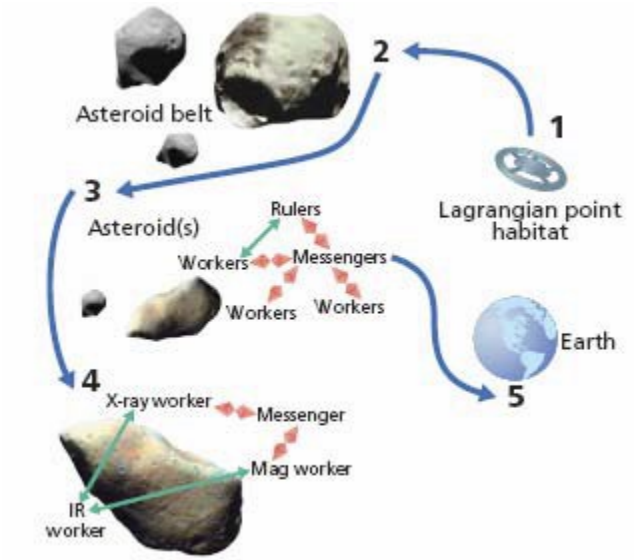
Tieteellisen datan käsittelyyn on esitetty agenttipohjaista lähestymistapaa [Rilee et al., 2002]. Havaintovälineet tuottavat niin suuren määrän dataa, että tutkimuksen tehostamiseksi sen muokkaaminen paikan päällä olisi järkevää. Normaalisti avaruusprojekteissa tutkimusdata käsitellään ja muokataan tieteelliseen muotoon maassa ihmisten avustuksella, jotta tutkimuksen tieteellisestä arvosta voidaan olla varmoja. Samat toiminnot pitäisi nyt pystyä siirtämään tuotettavaksi paikan päällä avaruusaluksissa ja vain lopputulokset lähetetään maahan. Tähän tehtävään NASA on kehittänyt älykästä Science Agent (SA) -ympäristöä.

Edellä esitetyn toiminnan rakentaminen parveen, jotta se kehittyisi ja sopeutuisi tehtävän edetessä, on tärkeä tutkimuksen kohde. Tutkittavia lähestymistapoja tekoälyn toteuttamiseksi ovat mm. neuroverkot, sumea logiikka, hajautettu tekoäly ja geneettiset algoritmit.

4.2. Prospecting ANTS Mission (PAM)

NASAn ANTS/PAM on projekti, joka tähtää mahdollisimman autonomiseen asteroidivyöhykkeen tutkimukseen [Curtis et al., 2000]. Asteroidivyöhyke sijaitsee Marsin ja Jupiterin kiertoratojen välissä ja sen etäisyys auringosta on 2-4 AU. Vyöhykkeellä arvioidaan olevan noin 1,5 miljoonaa kohdetta, jotka ovat läpimitaltaan suurempia kuin 1 km [Stenger, 2002]. Eräiden arvioiden mukaan 100 m - 1 km läpimitaltaan olevien kohteiden keskimääräinen etäisyys toisistaan olisi noin 100 000 km. Tavoitteena on tehdä tieteellinen kartoitus kaikista yli 1 km halkaisijaltaan olevista asteroideista. Kartoitus auttaa löytämään sekä maassa hyödyllisiä että avaruuden tutkimisessa hyväksikäytettäviä luonnonvaroja.

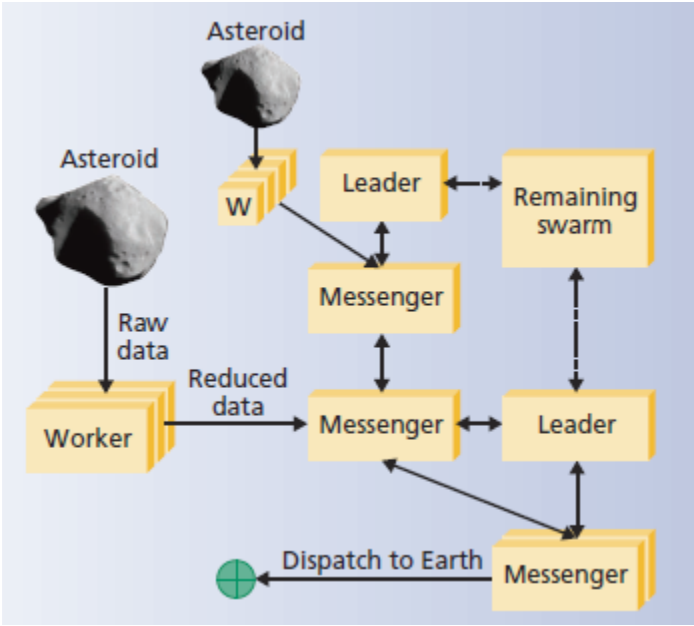
Tutkimus on tarkoitus toteuttaa 1000 pienen satelliitin avulla. NASAn Goddard Space Flight Center (GSFC) kehittää alle 10 kg painavaa avaruusalusta sekä näiden yhteistoimintaa [Panetta et al., 1998]. Yksittäisen satelliitin massa on tarkoitus saada alle yhden kilogramman. Satelliitit lähetään matkaan maata kiertävältä radalta aurinkopurjeiden avulla yhtenäisessä muodostelmassa kohti asteroidivyöhykettä. Jos käytössä on aurinkopurje, jonka pinta-ala on 100 m^2, niin matka taittuu noin 3,5 vuodessa. Tarpeen tullen maasta voidaan lähettää uusia satelliitteja paikkaamaan menetettyjä projektin ollessa jo käynnissä. Kuva 4 havainnollistaa asteroidien tutkimusta.



Kuva 4. Havainnekuva asteroidien tutkimuksesta

Lähetettävät satelliitit jakautuvat kolmeen luokkaan: Worker, Messenger ja Ruler. Suurin osa satelliiteista, 80–90 %, on ns. työläisiä (Worker) ja lopuilla pääosaaminen liittyy joko viestintään (Messenger) tai tehtävien johtamiseen (Ruler). Asteroidivyöhykkeelle päästyään osa satelliiteista jää etäämmälle asteroideista, kun taas osa asettuu kiertoradalle asteroidien ympärille suorittamaan mittauksia tai painovoiman niin salliessa leijumaan asteroidin läheisyyteen.

Jokaisessa satelliitissa on agenttipohjaisia tekoälyominaisuuksia liittyen ainakin tutkimukseen, navigointiin ja operaatioihin. Tutkimusagentti pitää huolen tieteellisistä havainnoista ja ohjaa niitä varten tehtäviä päätöksiä. Navigaattori huolehtii navigoinnista ottaen huomioon erilaiset vaaratekijät sekä muutokset esimerkiksi aurinkotuulen voimakkuudessa ja aurinkopurjeen toiminnassa. Operaatioista vastaava agentti on kuin ihminen lennonvalvonnassa eli se hoitaa tehtävää koskevien isojen päätösten tekemisen. Päätöksenteon pohjaksi on annettu ohjeita projektin alussa, mutta myös erilaisilla koneoppimismenetelmillä saatu lisätieto tehtävän aikana vaikuttaa päätöksentekoon. Operaattori pitää huolen mm. parven työtehtävistä, reitin suunnittelusta, hakumenetelmien optimoinnista, kommunikaatiosta, tieteellisen datan organisoinnista ja resurssien hallinnoinnista. Myös erikoistuneita tietokonesiruja suunnitellaan käytettäväksi toimintojen helpottamiseksi [Curtis et al., 2000]. Kuvassa 5 havainnollistetaan tiedon kulkua tutkimuksen aikana.



Kuva 5. Tutkimustulosten kerääminen

Tekoäly ja heuristinen järjestelmä

Projektin arkkitehtuurin pitää olla mahdollisimman autonominen ja sen saavuttamiseksi suunnitellaan käytettäväksi heuristisia järjestelmiä. Heuristisilla menetelmillä tavoitellaan parempia tuloksia tutkimustehtävissä ja erilaisten operaatioiden tehokkuudessa. Heuristisen menetelmien toteuttamiseksi tutkitaan mm. neuroverkkojen, sumean logiikan, hajautetun tekoälyn ja geneettisten algoritmien käyttöä eri yhteyksissä.

Neuroverkkoja voidaan käyttää erilaisissa oppimistehtävissä sekä esimerkiksi asteroidien hahmontunnistuksessa. Neuroverkkojen ideaa voidaan hyödyntää koko satelliittiparven toteuttamisessa. Kun jokainen satelliitti on yksi neuroverkon neuroni, niin vikasietoisuus paranee huomattavasti eikä yksittäisen satelliitin menetys tee suurtakaan vahinkoa parven toiminnalle.

Sumean logiikan sovelluksilla saadaan hyötyä satelliittien ohjaustoiminnoissa ja konfliktien ratkaisussa. Sumeaa logiikkaa voidaan käyttää, kun arvioidaan asteroidin tärkeyttä tutkimuskohteena, jolloin kaikki halutut ominaisuudet asteroidista vaikuttavat valintaan. Toisena esimerkkinä voidaan käyttää röntgenspektrin laadun tutkimista. Monet tekijät vaikuttavat jälleen lopputulokseen ja tarvittaessa voidaan tehdä muutoksia, jotta spektrin laatu paranee.

Hajautettu tekoäly yhdistää tekoälytekniikoita ja ”multiple problem solvers” [Martial, 1992]. Sen avulla tuetaan päätöksentekoa ja parven kollektiivista älykkyyttä. Se sopii hyvin käytettäväksi ANTS-projekteissa, koska ne ovat luonteeltaan hajautettuja kuten tässä tapauksessa satelliittien parvi, joka on hajallaan asteroidivyöhykkeellä. Hajautetulla tekoälyllä voidaan hyödyntää parven tietokoneiden yhteistehoa jakamalla ongelma osiin.

Hajautettu tekoäly voidaan jakaa hajautettuun ongelmanratkaisuun ja multiagenttijärjestelmiin [Martial 1992]. Näistä jälkimmäinen soveltuu hyvin ANTS-projekteihin, koska jokainen satelliitti on vastuussa selviytymisestään, jokaisella satelliitilla on tietty tehtävä osana parvea ja jokaisen satelliitin

on kommunikoitava parven muiden satelliittien kanssa suoriutuakseen omasta tehtävästään. Kaikilla satelliiteilla on yhteisenä päämääränä kerätä tietoa asteroideista, mutta jokaisella yksilöllä on myös omia tavoitteita kuten energian säästäminen. Edellä mainittu on yksi multiagenttijärjestelmän tärkeimpiä ominaisuuksia.

Geneettiset algoritmit puolestaan tehostavat hakutoimintoja ja navigaatiota. Yleisesti ottaen ne ovat tehokkaita esimerkiksi optimointitehtävissä ja automaattisessa ohjelmoinnissa. Niitä käytetään myös etsittäessä ratkaisuja NP-vaikeisiin ongelmiin kuten kauppamatkustajan ongelma. PAM-projektissa geneettisiä algoritmeja voidaan käyttää esimerkiksi vahingoittuneen aurinkopurjeen säätämisessä. Vahingon tarkka analysointi voi olla vaikeaa, mutta sen vaikutusta voidaan minimoida geneettisen algoritmin avulla.

Heuristisia järjestelmiä varten suunnitellaan erityistä tiekonesirua, ”spacecraft on a chip”, joka pystyy erikoistuneilla menetelmillä tukemaan niin tutkimusvälineiden käyttöä kuin satelliittien tekoälyäkin. Sirun kehittämisellä tavoitellaan myös projektin kompleksisuuden vähenemistä, kulujen pienenimistä sekä parempaa suoritus- ja selviytymiskykyä.

Järjestelmän logiikka voidaan jakaa alempaan ja ylempään tasoon. Ylemmällä tasolla tapahtuu pääasiassa kaikki tehtävän suunnitteluun liittyvä toiminta. Alempi taso on vastuussa turvallisuusseikoista sekä pääjärjestelmistä kuten liikkuminen, kommunikointi ja havainnointi. Alemman tason toiminnot ovat aina käytettävissä, mutta ylempi taso voidaan sammuttaa esimerkiksi energian säästämisen vuoksi [Almeida et al., 2002].

Satelliitin liikkumisen hallinta on tärkeää paitsi törmäysten välttämiseksi myös kommunikoinnin kannalta. Satelliitit kykenevät kommunikoimaan keskenään vain, mikäli ne ovat oikeassa asennossa suhteessa toisiinsa. Tekoälyn täytyy osata havaita erilaisten sensorien ongelmat ja joko korjata ne tai ottaa mahdollinen varajärjestelmä käyttöön.

Yksi tärkeimmistä ja käyttökelpoisimmista tekoälyn ohjelmointikielistä on Lisp [ALU, 2001]. Lisp sisältää tekoälyohjelmointiin sopivia ominaisuuksia kuten tuen listojen käsittelyyn, hahmontunnistukseen ja ”exploratory”-ohjelmointiin. Se on ideaali valinta tekoälyn sovelluksille laajalle levinneen käyttönsä vuoksi. Lisp voisi olla hyvä valinta PAM-projektin toteuttamiseksi, koska se sopii erinomaisesti suurten, monimutkaisten ja kriittisten applikaatioiden ohjelmointiin.

Euroopan avaruusjärjestö ESA on järjestänyt vuodesta 2005 lähtien kilpailun nimeltä Global Trajectory Optimisation Competition (GTOC), jossa kilpaillaan erilaisten avaruustutkimukseen liittyvien lentoratojen suunnittelussa. Vuonna 2014 järjestetyssä seitsemännessä kilpailussa tehtävänä oli tutkia mahdollisimman monta asteroidia kolmella luotaimella. Tehtävän suorittamiseksi Maasta laukaistaan emäalus asteroidivyöhykkeelle. Perille päästyään emäalus lähettää matkaan kolme luotainta, joiden tehtävänä on käydä mahdollisimman monella noin 16000 asteroidista ja sen jälkeen palata takaisin emäalukselle. Tehtävän suunnittelussa pitää huomioida erilaiset rajoitteet mm. emäaluksen ja luotainten polttoaineen käytön suhteen sekä 12 vuoden kokonaisaika ja 6 vuotta/luotain.

Kilpailussa parhaaseen tulokseen ylsi NASAn Jet Propulsion Laboratory (JPL) vierailemalla 36 asteroidilla reilussa kymmenessä vuodessa. JPL käytti ratkaisussaan hyväksi Lambertin ongelman

ratkaisuja, muurahaiskoloniaoptimointia, hiukkasparvioptimointia, geneettisisä algoritmeja sekä epälineaarisen optimoinnin menetelmää ohjelmalla SNOPT. Kilpailussa hyväksytyn tuloksen sai 27 joukkuetta, jotka keräsivät 13–36 pistettä (asteroidia).

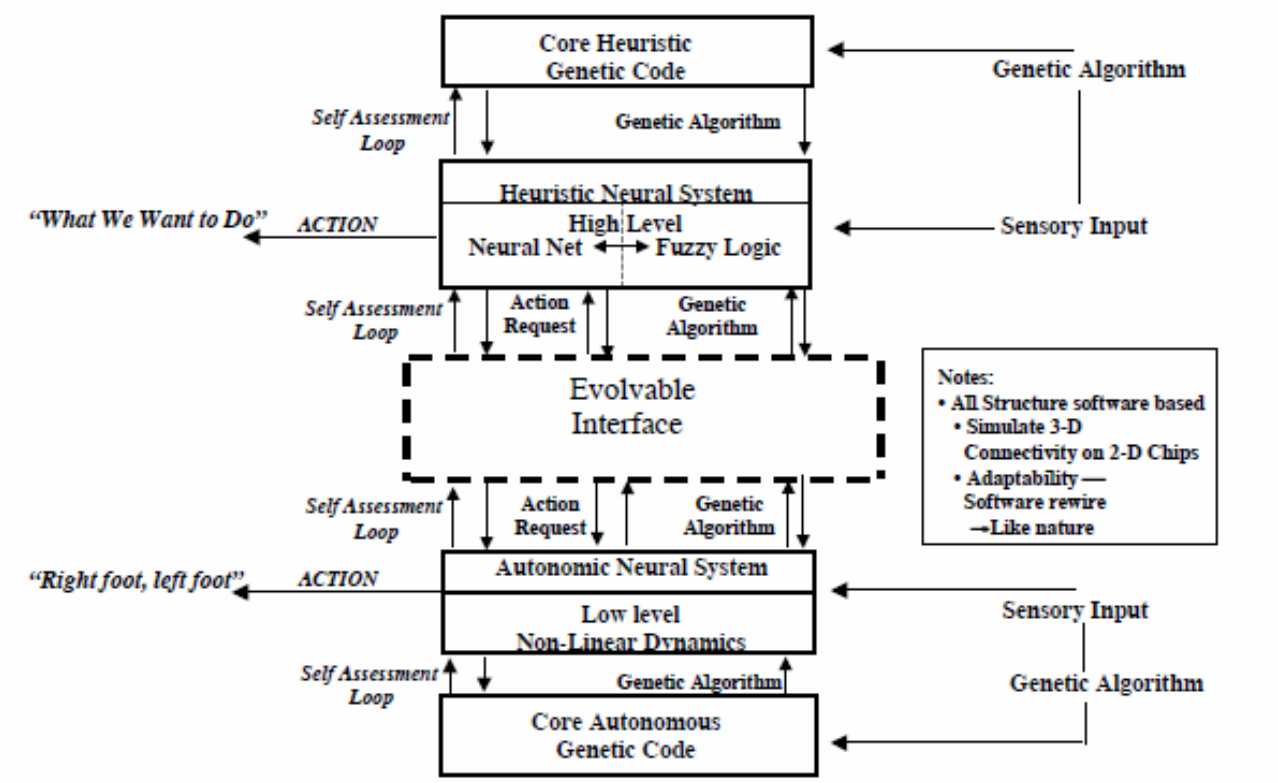
4.3. Hubble-avaruusteleskoopin autonominen pelastusoperaatio

NASAn Goddard Space Flight Center (GSFC) tekee tutkimus- ja kehitystyötä, jotta avaruustutkimus voisi olla aktiivista ja monimuotoista paikan päällä ennemmin kuin vain vierailuja avaruudessa kauko-ohjauksen avulla. Keskeisessä osassa on ollut Autonomous Nano-Technology Swarm (ANTS) -tehtäväarkkitehtuuri sekä Neural Basis Function/Synthetic Neural System (NBF/SNS) [Rilee et al., 2005]. Nämä järjestelmät ovat epälineaarisia ja skaalautuvia sekä suunniteltuja toimimaan erilaisten epävarmuustekijöiden vallitessa.

NBF/SNS-järjestelmä on yhdistelmä alemman ja ylemmän tason järjestelmätoimintoja. Erikoistuneet komponentit on sulautettu Evolvable Neural Interfacen eli ENIin, joka välittää informaatiota järjestelmän sisällä. Yksittäinen NBF/SNS vastaa yksinkertaisista toiminnoista, mutta monimutkaisemmissa linkitetään useampia NBF/SNS-yksiköitä ENIn välityksellä. ENIn tekee joustavaksi systeemiksi perustana oleva geneettinen kehittyminen ja harjoituttaminen, jonka avulla selvitään esimerkiksi NBF/SNS-komponenttien ristiriitojen ratkaisemisestä. Kehitteillä on järjestelmää demonstroiva Hubble-avaruusteleskoopin pelastusskenaario, jossa alemman tason toimintona oleva avaruudelliseen asentoon ja propulsioon liittyvä ohjausjärjestelmä sekä ylemmän tason toimintona navigoinnin johtaminen toimivat yhdessä.

Biologisissa systeemeissä suuri osa ohjauksesta tapahtuu autonomisella tasolla ilman tietoista ohjausta, kuten hengitys, sydämen syke ja tasapaino. Samanlaiseen toimintaan pyritään tässä tapauksessa Mark Tildenin kehittämällä ”nervous nets” -neurojärjestelmällä [Rilee et al., 2004]. Kyseisessä järjestelmässä sensoreiden tieto välittyy ”nervous nets” -osaan, joka lähettää ohjaussignaaleita aktuaattoreille esimerkiksi tasapainoa tai liikkumista varten. Neuroverkon robustisen luonteen ansiosta järjestelmä ylläpitää haluttua tasapainotilaa tai liikettä automaattisesti.

Evolvable Neural Interface on osa arkkitehtuuria, jossa alemman tason komponentit kommunikoivat ylemmän tason komponenttien kanssa sen välityksellä. ENI on kehittyvä, koska se osaa sopeuttaa signaalien kommunikointiansa tehtävän edetessä. Neuraalisuus näkyy neuroverkon omaisena kolmiulotteisena kommunikaatioverkkona, jonka solmut saattavat myös modifioida signaaleja. Yhdessä alemman ja ylemmän tason komponenttien kanssa ENI muodostaa Neural Basis Function -arkkitehtuurin (NBF) (kuva 6).



Kuva 6. NBF-arkkitehtuurin perusrakenne

NBF-arkkitehtuurissa siis sekä alemman että ylemmän tason komponentit syöttävät informaatiota ENI:lle. Molemmilla tasoilla komponentit taas saavat informaatiota sekä sensoreilta että ENI:ltä. Lopputuloksena järjestelmä tuottaa tarkoituksellista toimintaa. Kehitysvaiheessa ENI opetetaan simulaatioiden avulla oppivaksi järjestelmäksi, jossa ylemmän tason abstrakteja toimintoja liitetään alemman tason käyttäytymismalleiksi. NBF-systeemin sopeutumiskykyisyyden vuoksi valmiiseenkin järjestelmään voidaan lisätä uusia, johonkin osa-alueeseen erikoistuneita NBF-elementtejä ja näin parantaa järjestelmän toimivuutta.

Skaalautuvuuden avulla NBF on käyttökelpoinen järjestelmän monilla eri tasoilla. Alijärjestelmillä on omat NBF:nsä ja yhteistyö muiden alijärjestelmien kanssa hoituu ENIn kautta. Esimerkiksi monitasoisessa ja satoja avaruusaluksia sisältävässä ANTS/PAM-projektissa ENI on läsnä koko parvessa, jossa NBF-alijärjestelmien tai avaruusalusten tasolla voi tuotta alemmas tason autonomisia toimintoja parven osille. Järjestelmän kehittyminen ja adaptoituminen projektin aikana on suuri edistysaskel perinteisestä avaruustutkimuksesta. Näillä ominaisuuksilla NBF:iin perustuva Synthetic Neural System (SNS) vähentää koko järjestelmän kompleksisuutta ja helpottaa sen organisointia sekä kontrollointia.

Sovellettaessa SNS-arkkitehtuuria Hubble-avaruusteleskoopin pelastamiseksi oletettiin teleskoopin pahimmassa tapauksessa liikkuvan avaruudessa hallitsemattomasti ja pyörivän oman akselinsa ympäri jopa kaoottisen ennustamattomasti. Teleskoopin liikkeeseen vaikuttaa myös ulkopuolisia voimia.

Simulaatioilla harjoitettiin NBF-ohjausohjelmistoa sopeutumaan erilaisiin vaihtoehtoihin teleskoopin liikkeessä ja varsinkin kaikkein äärimmäisiin olosuhteisiin. Tildenin työhön perustuvan synteettisten neuroneiden verkon avulla suoritettiin alemman tason kontrollointi pelastusaluksen ohjaamiseksi. Järjestelmän ENI perustui perinteisempään keinotekoiseen neuroverkoon, mutta sen harjoittamiseen oli käytetty kehittynyttä Kalman-suodatinta [Lary et al., 2004a; 2004b]. Alustavien simulaatioiden perusteella ENI kykenee antamaan riittäviä ohjeita pelastusaluksen alemman tason liikettä ohjaaville komponenteille perustuen ylemmän tason käskyihin sekä sensoreiden informaatioon pelastusaluksen ja teleskoopin liikkeistä. Lisäksi selvisi, että ENIn neuroverkko oppi ennakoimaan teleskoopin liikkeitä riittävällä tarkkuudella, jotta pelastusaluksen reaaliaikainen ohjaaminen ja teleskoopin pelastaminen vangitsemalla on mahdollista. Nämä alustavat tulokset avaavat tietysti uusia ja mielenkiintoisia mahdollisuuksia tuleville avaruusprojekteille.

4.4. Precision Formation Flying

NASAn Jet Propulsion Laboratoryn yhtenä tutkimuskohteena on Precision Formation Flying, jonka tavoitteena on kehittää ohjausjärjestelmiä avaruutta yhteistyössä tutkiville avaruusaluksille. Kehitettävät komponentit kuuluvat arkkitehtuuriin, metodologioihin, laitteistoon ja ohjelmistoon. Yhtenä tavoitteena on kehittää muodostelmiin liittyviä metodeja, arkkitehtuuria, algoritmeja ja ohjelmistoja, jotta muodostelmassa liikkuminen ja sen ylläpitäminen olisi mahdollista. Järjestelmiltä vaaditaan suurta tarkkuutta, kestävyyttä ja tehokkuutta.

Muodostelmassa lentämisellä tarkoitetaan avaruusalusten joukkoa, joka kykenee vuorovaikutteiseen yhteistoimintaan. Toiminta määritellään yhteisellä ohjausjärjestelmällä. Muodostelmassa lentämisen ohjaaminen on kuuden vapausasteen (6 DOF) liikkeen ongelma ja muodostelman vaihtaminen tai uuteen tilaan siirtyminen on kaikkein monimutkaisin operaatio. 6 DOF käsittää avaruusaluksen vapaan liikkumisen kolmiulotteisessa avaruudessa. Liikesuuntia ovat 1) eteen ja taakse, 2) oikealle ja vasemmalle, 3) ylös ja alas. Pyörimissuuntia puolestaan ovat 4) vaakasuunta, 5) pystysuunta ja 6) sivusuunta. Myös liikkuminen kolmessa vapausasteessa on mahdollista, jolloin pyörimisiike jätetään pois.

Käytetään seuraavassa esimerkkinä Terrestrial Planet Finderia, jossa kaksi tai useampia teleskooppeja heijastavat tutkimuskohteen kolmannelle alukselle, joka yhdistää tutkimustiedot. Tällainen toiminta vaatii teleskoopeilta samanaikaisia liikkeitä, jolloin toimintojen suorittamiseen vaadittavat laskutoimitukset kasvavat sitä suuremmiksi, mitä useampia teleskooppeja on käytössä. Parempi menetelmä nykyaikana on hyödyntää hajauttamista ja moniprosessorijärjestelmiä. Lisäksi pitää pystyä parantamaan mittatarkkuuksia. Perinteisillä teknologioilla päästään tyypillisesti luokkaa 10^9 ole-

vaan tarkkuuteen, kun Terrestrial Planet Finder vaatii 10^{11} mittatarkkuutta. Jotta synkronoidut mittaukset onnistuvat, on muodostelman pysyttävä lentoradallaan ±1 cm tarkkuudella alusten etäisyyksien vaihdellessa 20 m ja 100 m välillä.

Muodostelmien kontrolloinnissa ja ohjauksessa on tärkeää kiinnittää huomiota liikkeen tarkkuuteen, törmäysten estämiseen, polttoaineen kulutuksen minimoimiseen ja tasapainottamiseen sekä havainnointi- ja kommunikaatiotarpeiden minimointiin.

Muodostelmien liikkeessä tärkeässä osassa on reaaliaikainen optimointi. Ohjaamiseen, kontrollointiin, arvioimiseen ja päätöstentekoon liittyvät ongelmat tarvitsevat nopeita ja vakaita optimointialgoritmeja. Näihin ongelmiin lukeutuvat lentoratojen suunnittelu polttoaineoptimaalisesti ja muodostelman pitäminen vakaana liikuttaessa paikasta toiseen.

Precision Formation Flying käyttää konveksiin optimointiin (COP) perustuvia globaaleja optimointimenetelmiä. Näillä menetelmillä ratkaisun löytäminen on mahdollista polynomisessa ajassa, esimerkkinä mainittakoon Interior Point Method (IPM). Niinpä monet ongelmat formuloidaan konveksin optimoinnin ongelmiksi ja erityisesti Semi-Definite Programming (SDP) tai Second-Order Cone Programming (SOCP) -ongelmiksi. Tällä tavoin on mahdollista löytää globaalisti optimaaliset ratkaisut. Lisäksi IPM:n deterministisen konvergenssin ominaisuudet tekevät siitä sopivan autonomiseen käyttöön avaruusaluksissa. Tästä syystä monet ohjaamiseen ja kontrollointiin liittyvät ongelmat kannattaa formuloida COP:lla tai varsinkin SOCP:lla ratkaistavaan muotoon [http://dst.jpl.nasa.gov/real-time_optimization/].

Valitettavasti esimerkiksi muodostelman uudelleenjärjestämisen ongelma ei ratkea konveksilla optimoinnilla. Tällaisille ei-konvekseille ongelmille voidaan kuitenkin soveltaa peräkkäisen konveksoinnin menetelmiä.

Açıkmese ja muut [2006] esittävät erään ratkaisun muodostelman uudelleenkonfiguroimiseksi. Siinä ohjauksen ongelma formuloidaan ensin joko jatkuvaksi polttoaineen tai energian käytön minimoinnin optimointiongelmaksi ottamalla lukuun myös yhteentörmäysten välttämisen sekä kontrollointiin liittyvät rajoitukset.

Seuraavassa vaiheessa optimointiongelma diskretisoidaan, jotta päästään äärelliseen määrään parametrien optimointiongelmia. Tässä muodossa saadaan määritettyä tasot avaruusalusten välille pitämään huolta yhteentörmäysten välttämisestä. Sitten heuristiikka erottelee ne avaruudelliset tasot, jotka johtavat törmäysten välttämiseen tarvittavien rajojen konveksisointiin. Lisäksi määritellään erikseen rajat, jotka takaavat, että aikaikkunoiden välissä ei tapahdu törmäyksiä. Tämä lisäys mahdollistaa pidemmät aikaikkunat ja täten lyhentää algoritmin suoritusaikaa.

Edellä esitetyn tuloksena saadaan äärellinen Second-Order Cone Programming (SOCP) optimointiongelma. Deterministisen konvergenssin ja ennakkoon määrätyllä tarkkuuden tasolla voidaan sitten perusalgoritmeilla (IPM) ratkaista ongelmalle globaali optimi. Tällainen algoritmi muodostelman uudelleenkonfiguroimiseksi soveltuu toteutettavaksi avaruusaluksessa reaaliaikaisia operaatioita varten.

Scharf ja muut [2003] ovat laajasti selvittäneet muodostelmassa lentämiseen kehitettyjä algoritmeja. Richardsin ja muiden [2002] menetelmä perustuu Mixed-Integer Linear Programming (MILP) -ongelman formulointiin. MILP on luonnostaan epädeterministinen polynomiaalisen ajan vaatima NP-täydellinen algoritmi ja skaalautuu eksponentiaalisesti avaruusalusten lukumäärän mukaan.

Lentoratojen uudelleenkonfiguroinnissa käytetään hyväksi niiden parametrisointia polynomeiksi [Singh and Hadaegh, 2001; Sultan et al., 2004a; 2004b]. Singh ja Hadaegh [2001] ratkaisevat törmäysten estämisen asteittaisen vähenemisen algoritmilla, kun taas Sultan ja muut [2004a; 2004b] rajoittavat lentoratoja kulkemaan määrättyjen apupisteiden kautta.

Frazzoli ja muut [2001] ratkaisevat saman ongelman käyttämällä satunnaistettuja hakuja yhdessä konveksisen ohjelmoinnin kanssa. Muita ratkaisumalleja ovat esittäneet Wang ja Hadaegh [1999], Campbell [2003], Li ja muut [2000], Kim ja muut [2003] sekä Boyd ja Vandenberghe [2004].

Robotiikan puolella yleisiä tekniikoita ovat potentiaaliin perustuvat polun etsinnät, mutta niiden suora kopioiminen avaruudessa sovellettavaksi ei ole mahdollista puutteellisen yhteentörmäysten välttämisen vuoksi. Kirjallisuus miehittämättömien lentokoneiden ja vedenalaisten alusten saralta voi myös tarjota hyödyllisiä tekniikoita muodostelmien hallitsemiseen [McLain et al., 2001; Smith et al., 2001].

Yleisesti ottaen muodostelmalentämisen lentoratojen suunnittelun hankalin osa on törmäysten välttäminen, jossa ongelman laatu on NP-täydellinen [Canny et al., 1988; Garcia and How, 2005]. Açıkmese ja muut [2006] esittävät tutkimuksessaan kaksi ongelman ratkaisua parantavaa seikkaa: 1) törmäysten välttämiseen tarvittavien ehtojen konveksointi heuristisesti jolloin 2) nämä ehdot ovat samalla voimassa myös aikaikkunoiden välillä.

5. Yhteenveto

Avaruus on ihmiselle vihamielinen ymäpristö. Sen tutkiminen on kuitenkin kiinnostanut ihmisiä jo pitkän aikaa. Jotta avaruutta päästäisiin kunnolla tutkimaan, pitää sinne lähettää tutkimuslaitteistoa rakettien avulla. Rakettien laukaisuun liittyy aina riski sen tuhoutumisesta, jolloin menetetään ihmishenkiä. Myös pelkkä avaruudessa oleskelu on ihmisten terveydelle vaarallista, eikä pitkäaikaisen altistumisen kaikkia terveysvaikutuksia vielä edes tiedetä. Osittain näistä syistä tehdään tutkimus- ja suunnittelutyötä autonomisen avaruustutkimuksen edistämiseksi.

Autonomisella avaruustutkimuksella tarkoitetaan avaruusalusten, satelliittien ja robottien itsenäisesti suorittamia tutkimustehtäviä. Näissä tehtävissä ihmisten suorittaman ohjauksen osuus on minimoitu ja tehtävän aikana tarvittavat muutokset suunnitellaan ja suoritetaan paikallisesti tekoälyn avulla.

Perinteisesti avaruudessa tapahtuvaa tutkimusta varten on matkaan lähetetty yksi suuri avaruusalus, joka on sisältänyt kaiken tarvittavan tutkimuslaitteiston ja vaatinut voimakkaan laukaisuraketin päästäkseen pois maan vetovoimasta. Uusissa tutkimuksissa on selvitetty pienten nanokokoisten satelliittiryhmien toimivuutta sekä hyötyjä suureen avaruusalukseen verrattuna. Pienetkin satelliitit pitää tietysti laukaista raketin avulla avaruuteen, mutta avaruudessa niillä on paljon etuja suureen avaruusalukseen. verrattuna. Pienistä satelliiteista jokainen on erikoistunut tiettyyn tutkimuksen osa-alueeseen ja omaa vain siihen tarvittavaa laitteistoa. Jos yksi satelliitti tuhoutuu tai menee epäkuntoon, on se helppo korvata toisella samanlaisella ryhmään kuuluvalla satelliitilla. Jos taas yhden suuren avaruusaluksen tutkimustehtävässä jokin laitteisto menee epäkuntoon, ei sitä ole yhtä helppo korvata ja pahimmassa tapauksessa koko tutkimustehtävä vaarantuu.

NASA on suunnitellut ANTS-arkkitehtuuria huolehtimaan satelliittiparvien toiminnasta. Heidän suunnitelmissaan tutkimustehtävää voisi kokonaisuudessaan olla suorittamassa jopa tuhat pientä satelliittia, jotka voivat jakautua tehtävän mukaisesti pienempiin ryhmiin. NASA:n suunnitelman mukaan satelliittiparven pitäisi olla

- itsestään konfiguroituva, eli kykenee sopeutumaan muutoksiin
- itsestään optimoituva, eli kykenee parantamaan toimintaansa
- itsestään parantuva, eli kykenee palautumaan virheistä ja laitteistongelmista sekä
- itseään suojeleva, eli kykenee ennakoimaan ja välttämään erilaisia uhkia.

ANTS-arkkitehtuurin mukainen parvi toimii joustavammin, luotettavammin ja autonomisemmin kuin yksittäinen suuri avaruusalus. Parven muita hyviä ominaisuuksia ovat responsiivisuus, mukautuvuus ja skaalautuvuus. ANTS-arkkitehtuurin avulla uskotaan myös kokonaiskustannusten laskevan mm. maasta suoritettavan tehtävän kontrolloimiseen ja ohjaamiseen tarvittavan työmäärän vähentymisen vuoksi. Kunhan satelliitit on ensin saatu lähtöpaikalleen maata kiertävälle radalle, on niiden sieltä tutkimustehtävään lähettäminen myös yksinkertaisempaa kuin maan avaruuskeskuksesta, jolloin jo käynnissä olevia tehtäviä voidaan täydentää ja mahdollisesti myös parantaa lähettämällä matkaan uusia satelliiteja.

ANTS-arkkitehtuurin toteuttamiseen on suunniteltu käytettävän erilaisia tekoälymenetelmiä, kuten geneettisiä algoritmeja, neuroverkkoja ja sumean logiikan menetelmiä. Tekoälymenetelmillä saadaan aikaan riittävä älykkyys parven itsenäiseen toimintaan. Tekoälymenetelmillä voidaan tehokkaasti hoitaa myös erilaisia optimointia vaativia tehtäviä, kuten lentoratojen laskemista, muodostelmien toteuttamista ja työtehtävien koordinoimista.

Siirtyminen deterministisistä stokastisiin järjestelmiin vaatii uudenlaista ajattelutapaa myös avaruuskeskuksissa, joissa avaruustutkimusta suoritetaan ja suunnitellaan, mutta se on välttämätön askel, kun tavoitellaan monimutkaisempia ja sopeutuvampia järjestelmiä. Toisaalta, ANTS on vasta suunnitteluvaiheessa ja ohjelmistokehityksessä on tapahduttava useita läpimurtoja ennen kuin kaikki kaavaillut ominaisuudet älykkyyden, autonomisuuden ja yhteistoiminnan suhteen pystytään toteuttamaan. Myös muilla osa-alueilla, kuten energian tuotannossa ja sen varastoinnissa sekä avaruusaluksen tai satelliitin liikkeen tuottavan raketin, aurinkopurjeen tai vastaavan voimanlähteen teknologiassa, on tapahduttava edistystä.

NASA on suunnitellut ANTS-arkkitehtuuriin sovellettavaksi mm. asteroidivyöhykkeen ja Saturnuksen kehien tutkimuksessa, Marsin pinta olisi mahdollista kartoittaa nopeasti pienillä miehittämättömillä lentokoneilla ja Kuuta voitaisiin tutkia sen pinnan muotoihin mukatauvilla tetrahedron-roboteilla, jotka perustuvat SMART-teknologiaan [Truszkowski et al., 2009].

ANTS-arkkitehtuuria on mahdollista hyödyntää myös kaivostoiminnassa ja vedenalaisessa tutkimuksessa sellaisissa paikoissa, jotka ovat ihmisille joko liian vaarallisia tai ahtaita. Sovelluksia löytyy varmasti myös aseteollisuuden puolelta esimerkiksi tiedustelutehtävistä. Myös GPS-paikannusta ja sen tarkkuutta voidaan parantaa aikaisempaa pienempien ja halvempien satelliittien avulla.

Viiteluettelo

[Abdelkhalik and Mortari, 2007] Abdelkhalik, Ossama, and Daniele Mortari, N-impulse orbit transfer using genetic algorithms, *Journal of Spacecraft and Rockets* 44, 2, 456-460, (2007).

[Açıkmese et al., 2006] Behçet Açıkmese, Daniel P. Scharf, Emmanuell A. Murray, and Fred Y. Hadaegh, A convex guidance algorithm for formation reconfiguration, In: *Proc. of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, (2006).

[Almeida et al., 2002] Shane Almeida, Ethan Croteau and Jonathan Freyberger, Autonomous Nano Technology Swarm (ANTS), Doctoral dissertation, Worcester Polytechnic Institute, (2002).

[ALU, 2001], What is Lisp? *The Association of Lisp Users (ALU)*, (2001). http://www.alu.org/.

[Beale and Jackson, 2010], Russell Beale and Tom Jackson, *Neural Computing - An Introduction*, CRC Press, (2010).

[Bellman, 1957] R. E. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, NJ. Republished 2003: Dover.

[Beni and Wang, 1989] Beni, G., and Wang, J., Swarm Intelligence in Cellular Robotic Systems, In: *Proc. NATO Advanced Workshop on Robots and Biological Systems*, Tuscany, Italy, June 26–30, (1989).

[Betts, 1998] Betts, John T., Survey of numerical methods for trajectory optimization, *Journal of Guidance, Control, and Dynamics*, 21, 2, 193-207, (1998).

[Bo and Feng, 2011] Bo, Xu, and Quansheng Feng, Research on constellation refueling based on formation flying, *Acta Astronautica* 68, 11, (2011).

[Bryson, 1975] Bryson, A. E., *Applied Optimal Control: Optimization, Estimation and Control*, CRC Press, (1975).

[Buckles and Petry, 1992] Buckles, Bill P., and Frederick Petry, *Genetic Algorithms*, IEEE Computer Society Press, (1992).

[Canny et al., 1988] Canny, J., Reif, J., Donald, B., and Xavier, P., On the Complexity of Kinodynamic Planning, In: *Proc. of 29th IEEE Annual Symp. on Foundations of Computer Science*, 306–316, (1988).

[Ceriotti and Massimiliano, 2010] Ceriotti, Matteo, and Massimiliano Vasile, An Ant System algorithm for automated trajectory planning, In: *Proc. of Congress on Evolutionary Computation (CEC)*, 2010, IEEE.

[Ceselli et al., 2012] Ceselli, A., G. Righini, and E. Tresoldi, Optimal Routing of Planetary Surface Exploration Vehicles, *Acta Futura* 5, 17-22, (2012).

[Coulouris et al., 2011] Coulouris, George, Jean Dollimore, Tim Kindberg and Gordon Blair, *Distributed Systems: Concepts and Design*, (5th Edition), Addison-Wesley, (2011).

[Curtis et al., 2000] Curtis, S., J. Mica, J. Nuth, and G. Marr., ANTS (Autonomous Nano Technology Swarm): An Artificial Intelligence Approach to Asteroid Belt Resource Exploration, In: *Proc. of 51st International Astronautical Congress*, (2000).

[Curtis et al. 2003a] Curtis, Steven A., Walt Truszkowski, Michael L. Rilee, and Pamela E. Clark, Ants for human exploration and development of space, In: *Proc. of Aerospace Conference*, 2003 IEEE, 1, 1-261, (2003).

[Curtis et al. 2003b] Curtis, S. A., M. L. Rilee, P. E. Clark, and G. C. Marr, Use of swarm intelligence in spacecraft constellations for the resource exploration of the asteroid belt, In: *Proc. of the Third International Workshop on Satellite Constellations and Formation Flying*, 24-26. (2003).

[Dachwald, 2004] Dachwald, Bernd, Optimization of Interplanetary Solar Sailcraft Trajectories Using Evolutionary Neurocontrol, *Journal of Guidance, Control, and Dynamics*, 27, 1, 66–72, (2004).

[Dachwald and Seboldt, 2002] Dachwald, Bernd, and Wolfgang Seboldt, Optimization of interplanetary rendezvous trajectories for solar sailcraft using a neurocontroller, In: *Proc. of AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, San Francisco, USA. (2002).

[Danoy et al., 2012] Danoy, Grégoire, Bernabe Dorronsoro, and Pascal Bouvry, New State-Of-The-Art Results for Cassini2 Global Trajectory Optimization Problem, *Acta Futura* 5, 65-72, (2012).

[Dantzig, 1947] Dantzig, G.B., Maximization of a linear function of variables subject to linear inequalities, 1947. 339–347 In: T.C. Koopmans (ed.): *Activity Analysis of Production and Allocation*, Wiley & Chapman-Hall, New York-London (1951).

[Davis and Lawrence, 1991] Davis and Lawrence, ed., *Handbook of Genetic Algorithms*, 115, Van Nostrand Reinhold, (1991).

[Dorigo et al., 1991] Dorigo, M., V. Maniezzo, and A. Colorni, *Positive feedback as a search strategy*, Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy, (1991).

[Dorigo, 1992] Dorigo, M., *Optimization, Learning and Natural Algorithms (in Italian)*, PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy, (1992).

[Dorigo et al., 1996] Dorigo, M., V. Maniezzo, and A. Colorni, Ant System: Optimization by a colony of cooperating agents, *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 26(1):29–41, (1996).

[Dorigo and Gambardella, 1997] Dorigo, M., and L. M. Gambardella, Ant Colony System: A cooperative learning approach to the traveling salesman problem, *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, (1997).

[Dorigo, 2007] Dorigo, M., Ant colony optimization, *Scholarpedia*, 2(3):1461, (2007).

[Dorigo et al., 2008] Dorigo, M., Oca, M. A. Montes de, and Engelbrecht, A., Particle swarm optimization, *Scholarpedia*, 3(11):1486, (2008).

[Duan et al., 2013] Duan, Haibin, Qinan Luo and Guanjun Ma, Hybrid Particle Swarm Optimization and Genetic Algorithm for Multi-UAV Formation Reconfiguration, *IEEE Computational Intelligence Magazine*, 8(3): 16-27 (2013).

[Eberhart and Kennedy, 1995] Eberhart, Russ C., and James Kennedy, A new optimizer using particle swarm theory, In: *Proc. of the Sixth International Symposium on Micro Machine and Human Science*, 39–43 (1995).

[Frazzoli et al., 2001] Frazzoli, E., Mao, Z., Oh, J., and Feron, E., Resolution of Conflicts Involving Many Aircraft via Semidefinite Programming, *AIAA Journal of Guidance and Control*, 24, 1, (2001), 79–86.

[Garcia and How, 2005] Garcia, I. and How, J., Trajectory Optimization for Satellite Reconfiguration Maneuvers with Position and Attitude Constraints, In: *Proc. of the American Control Conference*, 2005. IEEE, (2005).

[Gondzio and Terlaky, 1996] Gondzio, Jacek and Terlaky, Tamás, 3 A computational view of interior point methods, In: J. E. Beasley, *Advances in Linear and Integer Programming.* Oxford Lecture Series in Mathematics and its Applications 4. Oxford University Press. 103–144, (1996).

[Giron-Sierra and Ortega, 2002] Giron-Sierra, Jose M., and Guillermo Ortega, A survey of stability of fuzzy logic control with aerospace applications, *World Congress*, 15. 1. (2002).

[Hansen et al., 2003] Hansen, N., S.D. Muller, and P. Koumoutsakos, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES), *Evolutionary Computation*, 11, 1, 1–18, (2003).

[Hinchey et al., 2005] Hinchey, Michael G., James L. Rash, Walter F. Truszkowski, Autonomous and Autonomic Swarms, *Software Engineering Research and Practice*, 36-44, (2005).

[Holland, 1975] Holland, John H., *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University Michigan Press, (1975).

[Iacopino et al., 2014] Iacopino, C., Palmer, Ph., Policella, N., Donati, A., and Brewer, A., How Ants Can Manage Your Satellites, *Acta Futura*, 9, 59-72, (2014).

[Islam et al., 2012] Islam, Sk Minhazul, Swagatam Das, Saurav Ghosh, Subhrajit Roy, and Ponnuthurai Nagaratnam Suganthan, An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization, *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 42, 2, 482-500, (2012).

[Jones et al., 1993] Jones, D.R., C.D. Perttunen, and B.E. Stuckman, Lipschitzian optimization without the Lipschitz constant, *Journal of Optimization Theory and Applications*, 79, 1, 157–181, (1993).

[Kai and Ming, 2014] Kai, Y. and Ming, X., GPU Accelerated Genetic Algorithm for Low-thrust GEO Transfer Maneuvers, *Acta Futura*, 9, 49-57, (2014).

[Kamien and Schwartz, 2012] Kamien, M. I., and N. L. Schwartz, *Dynamic Optimization: The Calculus of Variations and Optimal Control in Economics and Management*, Courier Corporation, 2012.

[Karr and Freeman, 1997] Karr, Charles L. and L. Michael Freeman, Genetic-algorithm-based fuzzy control of spacecraft autonomous rendezvous, *Engineering Applications of Artificial Intelligence* 10, 3, 293-300, (1997).

[Kasabov, 1996] Kasabov, Nikola K., *Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering*. Marcel Alencar, (1996).

[Kephart and Chess, 2003] Kephart, Jeffrey O., and David M. Chess, The vision of autonomic computing, *Computer* 36, 1, (2003), 41-50.

[Kloetzer and Belta, 2007] Kloetzer, M. and Belta, C., Temporal logic planning and control of robotic swarms by hierarchical abstractions, *IEEE Transactions on Robotics*, 23, 320-330, (2007).

[Krieger et al., 2000] Krieger, M., J.-B. Billeter, and L. Keller, Ant-like task allocation and recruitment in cooperative robots, *Nature*, 406, 992, 31 August (2000).

[Ferrel, 1995] Ferrel, C., Global Behavior via Cooperative Local Control, *Autonomous Robots*, 2:2, 105-125, (1995).

[Laitinen, 2014] Laitinen, Erkki, Optimointiteoria, luentomoniste, Oulun yliopisto, (2014).

[Lary et al., 2004] Lary, D., Müller, M. and Mussa, H., Using neural networks to describe tracer correlations, *Atmospheric Chemistry and Physics*, 4, 143-146, (2004).

[Lary and Mussa, 2004] Lary, D., and Mussa, H., Using an extended Kalman filter learning algorithm for feed-forward neural networks to describe tracer correlations, *Atmospheric Chemistry and Physics Discussions*, 4, (2004), 3653-3667.

[Liang et al., 2006] Liang, Lily R., Shiyong Lu, Xuena Wang, Yi Lu, Vinay Mandal, Dorrelyn Patacsil, and Deepak Kumar, FM-test: A Fuzzy-Set-Theory-Based Approach to Differential Gene Expression Data Analysis, *BMC Bioinformatics*, 7 (Suppl 4): S7, (2006).

[Lin et al., 2012] Lin, Chien-Chou, Kun-Cheng Chen and Wei-Ju Chuang, Motion planning using a memetic evolution algorithm for swarm robots, *International Journal of Advanced Robotic Systems* 9, (2012).

[Martial, 1992] Martial, F. von, *Coordinating Plans of Autonomous Agents*, Springer-Verlag, (1992).

[McLain et al., 2001] McLain, Timothy W., Phillip R. Chandler, Steven Rasmussen and Meir Pachter, Cooperative control of UAV rendezvous, In: *Proc. of the 2001 American Control Conference*, 3, 2309-2314. IEEE, (2001).

[Michalewicz, 1992] Michalewicz, Zbigniew, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, (1992).

[Moscato, 1989] Moscato, Pablo, On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms, *Caltech Concurrent Computation Program, C3P Report* 826, (1989).

[Ohlmeyer and Phillips, 2006] Ohlmeyer, E.J. and Phillips, C.A., Generalized Vector Explicit Guidance, *Journal of Guidance, Control, and Dynamics*, 29, 2, 261-268, (2006).

[Olds et al., 2007] Olds, Aaron D., Craig A. Kluever, and Michael L. Cupples, Interplanetary mission design using differential evolution, *Journal of Spacecraft and Rockets* 44, 5, 1060-1070, (2007).

[Panetta et al., 1998] Panetta, P., Culver, H., Gagosian, J., Johnson, M., Kellogg, J., Mangus, D., Michalek, T., Sank, V., and Tompkins, S., NASA-GSFC Nano-Satellite Technology Development, In: *Proc. of the 12th AIAA/USU Conference on Small Satellites*, (1998).

[Pereira et al., 2008] Pereira, G. A. S., Kumar, V., and Campos, M. F. M., Closed loop motion planning of cooperating mobile robots using graph connectivity, In: *Proc. of Robotics and Autonomous Systems*, 56, 373-384, (2008).

[Phillips and Drake, 2000] Phillips, C.A., and J.C. Drake, Trajectory Optimization for a Missile Using a Multitier Approach, *Journal of Spacecraft and Rockets*, 37, 5, September–October (2000).

[Piotrowski et al., 2014] Piotrowski, Wiktor, Marcus Märtens, Dario Izzo, and Daniel Hennes, Space Hopper: a Serious Game Crowdsourcing the Design of Interplanetary Trajectories, *Acta Futura*, 9, 93-100, (2014).

[Pontryagin, 1987] Pontryagin, Lev S., *Mathematical Theory of Optimal Processes*, CRC Press, (1987).

[Price et al., 2006] Price, K.V., R.M. Storn, and J.A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, Springer, (2006).

[Rich and Knight, 1991] Rich, Elaine, and Kevin Knight. *Artificial Intelligence*. McGraw-Hill, Inc., (1991).

[Richards et al., 2002] Richards, A., T. Schouwenaars, J. How and E. Feron, Spacecraft Trajectory Planning with Avoidance Constraints Using Mixed-Integer Linear Programming, *AIAA Journal of Guidance and Control*, 25, 4, (2002), 755–764.

[Rilee et al., 2002] Rilee, Michael L., Scott A. Boardsen, Maharaj K. Bhatand Steven A. Curtis, Onboard science software enabling future space science and space weather missions, In: *Proc. of Aerospace Conference Proceedings*, 2002. IEEE, (2002), 2071-2084.

[Rilee et al., 2004] Rilee, M., S. Curtis, C. Cheung and J. Dorband, Evolving a Self-organizing Neuromechanical System for Self-healing Aerospace Structures, In: *Proc. of CANEUS 2004 Conference on Micro-Nano-Technologies for Aerospace Applications*, November, (2004).

[Rilee et al., 2005] Rilee, Michael L., Steven A. Curtis, John E. Dorband, Cynthia Y. Cheung, Jacqueline J. LeMoigne, David E. Lary and Hamse Y. Mussa, Working notes on a developmental approach to autonomous spacecraft and the recovery of the Hubble Space Telescope, In: *Proc. of the AAAI Spring Symposium on Developmental Robotics*, Stanford, CA, March 21-23, (2005).

[Rouff, 2007] Rouff, Christopher, Intelligence in Future NASA Swarm-based Missions, In: *Association for the Advancement of Artificial Intelligence, Fall Symposium 2007, Regarding the Intelligence in Distributed Intelligent Systems*, 112-115, (2007).

[Ross, 2009] Ross, I. M., *A Primer on Pontryagin's Principle in Optimal Control*, Collegiate Publishers, (2009).

[Ross and Karpenko, 2012] Ross, I. M., and M. Karpenko, A Review of Pseudospectral Optimal Control: From Theory to Flight, *Annual Reviews in Control*, 36, 182-197, (2012).

[Salo, 1994] Salo, Seppo, *Optimiohjausteoria ja variaatiolaskenta*, Helsingin kauppakorkeakoulun Kuvalaitos, (1994).

[Scharf et al., 2003] Scharf, D., Ploen, S., and Hadaegh, F., A survey of spacecraft formation flying guidance and control (part I): Guidance, In: *Proc. of the American Control Conference*, 2, American Automatic Control Council (2003).

[Schlueter, 2012] Schlueter, M., *Nonlinear mixed integer based optimization technique for space applications*, Diss. University of Birmingham, (2012).

[Silberschatz and Galvin, 1999] Silberschatz, Avi, and Peter Galvin, *Operating System Concepts*, John Wiley & Sons, Inc., (1999).

[Silvennoinen, 2004] Silvennoinen, Risto, Matemaattinen optimointiteoria 2, Tampereen teknillinen yliopisto, (2004).

[Silvennoinen, 2010] Silvennoinen, Risto, Matemaattinen optimointiteoria 1, Tampereen teknillinen yliopisto, (2010).

[Simoes et al., 2012] Simoes, L., Bourdarias, C., and Ribeiro, R.A., Real-Time Planetary Landing Site Selection - A Non-Exhaustive Approach, *Acta Futura*, 5, 39-52, (2012).

[Singh and Hadaegh, 2001] Singh, G., and Hadaegh, F., Collision Avoidance Guidance for Formation-Flying Applications, In: *AIAA Guid., Nav. & Contr. Conf.*, 2001.

[Sipper, 1996] Sipper, Moshe, A Brief Introduction to Genetic Algorithms, Last updated 2003. http://www.cs.bgu.ac.il/~sipper/ga.html.

[Smith et al., 2001] Smith, Troy R., Heinz Hanßmann and Naomi Ehrich Leonard, Orientation control of multiple underwater vehicles with symmetry-breaking potentials, In: *Proc. of the 40th IEEE Conference on Decision and Control*, 2001, 5, 4598-4603, (2001).

[Stenger, 2002] Stenger, Richard, Big asteroid population doubles in new census, CNN.com, 2002a. http://www.cnn.com/2002/TECH/space/04/05/asteroid.survey/index.html.

[Storn and Price, 1997] Storn, Rainer and Kenneth Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, *Journal of Global Optimization* 11.4: 341-359, (1997).

[Stracquadanio et al., 2011] Stracquadanio, Giovanni, Angelo La Ferla, Matteo De Felice, and Giuseppe Nicosia, Design of robust space trajectories, In: *Research and Development in Intelligent Systems* XXVIII, 341-354. Springer London, (2011).

[Stützle and Hoos, 2000] Stützle, T., and H. H. Hoos, MAX–MIN Ant System, *Future Generation Computer Systems*, 16(8):889–914, (2000).

[Sultan et al., 2004a] Sultan, C., Seereeram, S., Mehra, R. K., and Hadaegh, F. Y., Energy Optimal Reconfiguration for Large Scale Formation Flying, In: *Proc. of the American Control Conference*, June, 2004, 2986–2991, (2004).

[Sultan et al., 2004b] Sultan, C., Seereeram, S., and Mehra, R. K., Energy Optimal Multi-Spacecraft Relative Reconfiguration of Deep Space Formation Flying, In: *Proc. of the 43rd IEEE Conference on Decision and Control*, December, 2004, 284–289, (2004).

[Suominen, 2013] Suominen, Mikko, Parviälykkyys ja muurahaispohjaiset algoritmit, Pro gradu -tutkielma, Itä-Suomen yliopisto, Kuopio, (2013).

[Terlaky and Zhang, 1993] Terlaky, Tamás, and Shuzhong Zhang, Pivot rules for linear programming: A Survey on recent theoretical developments, *Annals of Operations Research*, 46, 1, 203–233, (1993).

[Tripp and Palmer, 2009] Tripp, Howard, and Phil Palmer, Distribution replacement for improved genetic algorithm performance on a dynamic spacecraft autonomy problem, *Engineering Optimization* 42, 5, 403-430, (2010).

[Tripp and Palmer, 2010] Tripp, Howard, and Phil Palmer, Stigmergy based behavioural coordination for satellite clusters, *Acta Astronautica* 66, 7, 1052-1071, (2010).

[Truszkowski and Hallock, 1999] Truszkowski, W., and Hallock, H., Agent Technology from a NASA Perspective, CIA-99, *Third International Workshop on Cooperative Information Agents*, Springer Verlag, Uppsala, Sweden, 31 July - 2 August (1999).

[Truszkowski et al., 2009] Walt Truszkowski, Lou Hallock, Christopher Rouff, Jay Karlin, James Rash, Michael G. Hinchey, and Roy Sterritt, *Autonomous and Autonomic Systems with Applications to NASA Intelligent Spacecraft Operations and Exploration Systems*, Springer, (2009).

[Vapnyarskii, 2011] Vapnyarskii, I.B., (originator), Bolza problem, *Encyclopedia of Mathematics*, URL: http://www.encyclopediaofmath.org/index.php?title=Bolza_problem&oldid=18748, (2011).

[Vapnyarskii, 2011] Vapnyarskii, I.B., (originator), Mayer problem, *Encyclopedia of Mathematics*, URL: http://www.encyclopediaofmath.org/index.php?title=Mayer_problem&oldid=17803, (2011).

[Vapnyarskii and Tikhomirov, 2011] Vapnyarskii, I.B., and V.M. Tikhomirov (originator), Lagrange problem, *Encyclopedia of Mathematics*, URL: http://www.encyclopediaofmath.org/index.php?title=Lagrange_problem&oldid=15919, (2011).

[Wang and Hadaegh, 1999] Wang, P., and Hadaegh, F., Minimum-Fuel Formation Reconfiguration of Multiple Free-Flying Spacecraft, *J. Astro. Sci*., 47, 1, 2, (1999), 77–102.

[Weiss, 1999] Weiss, Gerhard, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, (1999).

[Williams and Tucker, 1973] Williams, D.F., and W.B. Tucker, *Computation of Quasi-Optimal Reentry Trajectories Using the Simplex Algorithm of Linear Programming*, Report M-240-1208, Northrop Services, Inc., Huntsville, Alabama, April (1973).

[Xinsheng et al., 2006] Xinsheng, Ge, Zhang Qizhi and Chen Li-Qun, Optimal motion planning for a rigid spacecraft with two momentum wheels using Quasi-Newton method, *Acta Mechanica Solida Sinica*, 19, 4, 334-340, (2006).

[Xin-Sheng and Li-Qun, 2004] Xin-Sheng, Ge, and Li-Qun Chen, Attitude control of a rigid spacecraft with two momentum wheel actuators using genetic algorithm, *Acta Astronautica* 55, 1, 3-8, (2004).

[Xin-Sheng and Li-Qun, 2006] Xin-Sheng, Ge, and Li-Qun Chen, Optimal motion planning for non-holonomic systems using genetic algorithm with wavelet approximation, *Applied Mathematics and Computation* 180, 1, 76-85, (2006).

[Zadeh, 1965] Zadeh, L. A., Fuzzy sets, *Information and Control*, 8, 3, 338–353, (1965).

[Zou and Kumar, 2011] Zou, An-Min, and Krishna Dev Kumar, Adaptive fuzzy fault-tolerant attitude control of spacecraft, *Control Engineering Practice*, 19, 1, 10-21, (2011).